VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2014 Bc. Kamil Malík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra Informatiky

Kamerové testovací zařízení postavené na technologii PC

PC-based Camera Testing System

2014 Bc. Kamil Malík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Kamil Malík**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Kamerové testovací zařízení postavené na technologii PC
PC-based Camera Testing System

Zásady pro vypracování:

Při kontrole kvality sériové výroby se používají průmyslové kamery a techniky nazývané Strojové vidění. Průmyslová kamera spolu s vhodným objektivem a osvětlením sejme obraz výrobku a přenese jej po vhodném rozhraní do počítače. V počítači je pak nasnímaný obraz z kamery zpracován pomocí SW pro strojové vidění, který vyhodnotí kontrolované parametru výrobku.
Diplomová práce se zabývá kontrolou monochromatických řádkových/maticových LCD displejů bez podsvětlení. Kontrola spočívá v identifikaci zobrazených znaků/vzorů a kvantifikaci kvality zobrazení. Kvalita zobrazení souvisí s technologií výroby LCD, případně s poškozením displeje během operací při výrobě. Rychlost kontroly musí být do 1 sec. Cílem je vytvoření snímací scény, implementace algoritmu pro vyhodnocování kvality výroby,porovnání rychlosti navrženého algoritmu a implementovaného v LabVIEW, C/C++ a C#.
Postup prací:
1. Metodický přístup k řešení aplikací v oblasti průmyslového testingu. SW vývojová prostředí používaná pro automatizaci procesu měření - přehled a charakteristika existujících vývojových prostředí.
2. Rozbor nejčastěji používaných operací pro zpracování dvojrozměrného signálu (obrazu): srovnání se vzorem, detekce hran, konvoluční filtry atd., které budou následně využity při řešení diplomové práce.
3. Návrh měřícího zařízení, pro výše zadaný úkol. Vytvoření snímací scény včetně osvětlení, kamery a vyhodnocovacího systému. Vytvoření SW nástroje pro výpočet pracovní vzdálenosti objektivu v C/C++.
4. SW implementace výše definovaného jádra diplomové práce.
5. Porovnání časové náročnosti algoritmů implementovaných v jednotlivých vývojových prostředích.

Seznam doporučené odborné literatury:

[1] VLACH, Jaroslav, Josef HAVLÍČEK a Martin VLACH. Začínáme s LabVIEW. 1. vyd. Ilustrace Viktorie Vlachová. Praha: BEN - technická literatura, 2008, 247 s. ISBN 978-80-7300-245-9
[2] BITTER, Rick, Taqi MOHIUDDIN a Matt NAWROCKI. LabView advanced programming techniques. 2nd ed. Boca Raton: CRC Press, c2007, 499 s. ISBN 08-493-3325-3
[3] NI Vision Assistant Tutorial 2012, VA_Tutorial.PDF, součást instalace LabVIEW 2012
[4] NI Vision Builder for Automated Inspection Tutorial 2012, VBAI_Tutorial.pdf, součást instalace LabVIEW 2012

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Petr Bilík, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Čestné prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 6. 5.2014

Podpis

Poděkování

Děkuji vedoucímu své diplomové práce **doc. Ing. Petru Bilíkovi, Ph.D.** za ochotu, odbornou pomoc a cenné rady při zpracovávání této práce.

Abstrakt

Tato diplomová práce se zabývá kontrolou monochromatických řádkových LCD displejů pomocí techniky zvané strojové vidění. Cílem práce je vytvoření snímací scény, implementace algoritmů pro vyhodnocení kvality výroby technikou analýzy obrazu a porovnání rychlosti navržených algoritmů v LabVIEW a C/C++. Součástí práce je vytvoření nástroje pro výpočet pracovní vzdálenosti objektivu v C/C++.

Klíčová slova

Strojové vidění, NI LabVIEW 2013, NI Vision Acquisition Software, NI Vision Development Module, NI Vision Asistant, C++, Code::Blocks, wxSmith, wxWidgets, OpenCV, LCD 1602A

Abstract

This diploma thesis deals with machine vision test focused on monochromatic line LCD displays. The main goal of this work is to create camera scene, algorithms implementation for evaluation quality of production and compare the speed of proposed algorithms in LabVIEW and C/C++. Part of this work is creating of utility for calculation of work distance for camera lens in C/C++.

Key words

Machine vision, NI LabVIEW 2013, NI Vision Acquisition Software, NI Vision Development Module, NI Vision Asistant, C++, Code::Blocks, wxSmith, wxWidgets, OpenCV, LCD 1602A

Seznam použitých symbolů a zkratek

LCD - „Liquid Crystal Display“ displej z tekutých krystalů

USB - „Universal Serial Bus“ univerzální sériová sběrnice

PLC - „Programmable Logical Controller“ programovatelný logický automat

OCR - „Optical Character Recognition” optické rozpoznávání znaků

CCD - „Chanrge-Coupled Device“ zařízení s vázanými náboji

CMOS - „Complementary Metal-Oxide-Semiconductor“ doplňující se kov-oxid-polovodič

FPS - „Frame Per Second“ snímkovací frekvence

GigE - „Gigabit Ethernet“

POE - „Power Over Ethernet“ napájení po datovém síťovém kabelu

SW - „Software“ programové vybavení

VI - „Virtual Instrument“ virtuální přístroj, základní entita tvorby kódu v LabVIEW

subVI - „subroutine Virtual Instrument“ virtuální přístroj ve formě volané funkce

GUI - „Graphical User Interface“ uživatelské rozhraní

ROI - „Region Of Interest“ oblast zájmu

API - „Aplication Programing Interface“ rozhraní pro programování aplikací

Obsah:

1. Úvod ... 1
2. Hardware a teoretický úvod ... 2
2.1. LCD Displej 1602A ... 2
2.2. Technologie zobrazování LCD displeje 1602A ... 3
2.3. Připojení LCD displeje k PC ... 3
2.4. Použité PC a operační systém ... 5
2.5. Kamera a kamerové vybavení ... 5
2.5.1. Kamera ... 5
2.5.2. Objektiv ... 7
3. Vývojový software ... 11
3.1. NI LabVIEW ... 11
3.2. Podpůrný SW pro zpracování obrazu v NI LabVIEW ... 12
3.2.1. Vision Acquision Software ... 12
3.2.2. Vision Development Module ... 13
3.2.3. Vision Assistant ... 13
3.3. Nástroje pro programování a zpracování obrazu v jazyku C++ ... 14
3.3.1. Code::Blocks ... 14
3.3.2. GNU GCC ... 15
3.3.3. wxWidgets ... 16
3.3.4. OpenCv ... 16
4. Nástroj pro výpočet pracovní vzdálenosti objektivu ... 18
5. Snímací scéna a testovací snímky ... 19
5.1. Vytvoření snímací scény ... 19
5.2. Komunikace s displejem ... 20
5.3. Nasnímání Testovacích snímků ... 24
6. LCD Tester ... 27
6.1. Předpoklady pro správnou funkčnost vyhodnocovacího algoritmu ... 27
6.2. Kontrolované vlastnosti vyhodnocovacím algoritmem ... 27
6.3. Referenční korektní hodnoty pro kontrolované vlastnosti ... 28
6.4. Analýza pozadí LCD ... 29
6.5. Analýza pixelů LCD ... 30
6.5.1. Detekce pixelů hranovým detektorem ... 31

6.5.2. Detekce pixelů prahováním ... 33
6.5.3. Nástroj pro kontrolu LCD v LabVIEW ... 36
6.6. Algoritmy pro kontrolu LCD v C++ ... 39
6.6.1. Analýza pozadí ... 39
6.6.2. Analýza pixelů ... 40
7. Porovnání rychlosti algoritmů z jednotlivých vývojových prostředí ... 41
8. Závěr ... 42
9. Literatura ... 43
10. Seznam obrázků ... 45
11. Seznam tabulek ... 47
12. Seznam rovnic ... 47
13. Seznam příloh ... 47
14. Obsah přiloženého CD ... 47

1. Úvod

V současnosti je při sériové výrobě nejrůznějších průmyslových výrobků kladen velký důraz na důkladnou kontrolu kvality výroby a následně i samotného konečného výrobku. Existují specifické výrobky, kdy lze testy provádět výhradně formou optických testů. V takovémto případě je potřeba vizuálně kontrolovat stav, vzhled, barvu, rozměry a funkčnost vyráběného výrobku během jednotlivých částí výroby. Po dokončení výroby je potřeba finální výrobek důkladně ověřit. Při sériové výrobě již není z časových důvodů v lidských silách, každý výrobek zkontrolovat pomocí lidského zraku. Pro takové případy lze pak použít průmyslové kamery a techniku nazývanou strojové vidění případně průmyslové zpracování obrazu.

Zjednodušeně lze proces strojového vidění popsat následovně. Průmyslová kamera spolu s vhodným objektivem a osvětlením sejme obraz výrobku a přenese ho po vhodné komunikační sběrnici do počítače, nebo jiného logického zařízení určeného pro zpracování, jako je například PLC (Programmable Logical Controller). V počítači či PLC je následně nasnímaný obraz z kamery zpracován pomocí vhodného software. Program pro zpracování obrazu nasnímaný obraz dokáže vhodně upravit před dalším zpracování (např. zaostřit, vyfiltrovat, změřit jas a kontrast obrazu apod.) a poté provést vlastní vyhodnocení.

K základním úlohám při průmyslovém zpracování obrazu potom patří zejména:

- Měření rozměrů a vzdálenosti objektů
- Rozpoznávání barev v obraze a barevné korekce obrazu
- Nalezení hran objektů
- Počítání objektů v obraze a určení (klasifikace) typu objektů
- Rozpoznávání znaků (OCR) a čtení čárových a maticových kódů

Typická aplikace strojového vidění pak například může rozpoznávat přítomnost a správnou montáž všech komponent ve výrobku, změřit a zkontrolovat jeho rozměry a barvu, ověřit přítomnost a přečíst obsah štítku s čárový kódem, zkontrolovat povrchové vady (např. škrábance, deformace) a další kontroly.

Hlavním cílem této práce je vytvoření testovacího nástroje pro kontrolu kvality zobrazení a případného poškození monochromatických LCD displejů při výrobě. Algoritmus kontroly je navrhnut, s ohledem na co nejrychlejší vyhodnocení a tím je vhodný pro použití v sériové výrobě.

Mezi další cíle této práce patří, porovnání naimplementovaných algoritmů v různých vývojových prostředích, za použití v nich dostupných nástrojů pro zpracování obrazu a porovnání jejich časové náročnosti.

Jako testovaný výrobek byl vedoucím práce vybrán monochromatický dvouřádkový LCD displej 1602A.

2. Hardware a teoretický úvod

2.1. LCD Displej 1602A

Displej 1602A je monochromatický dvouřádkový maticový LCD (Liquid Crystal Display) displej sloužící pro zobrazování především textových informací, často využívaných v nejrůznějších zařízeních jako jsou prodejní automaty a jednočipová zařízení různého určení. Tento typ displeje se vyrábí v několika dalších variantách, které se liší počtem řádků, sloupců a velikostí matice znaku.

Každý řádek displeje 1602A tvoří 16 znaků, kde jeden zobrazovaný znak tvoří matice 5x8 čtvercových segmentů. Tento typ displeje je vybaven podsvícením, které tvoří zelená LED dioda nasvěcující pozadí za zobrazovanými znaky. Komunikaci a ovládání u displejů tohoto typu zajišťuje vestavěný řadič HD44780 od firmy Hitachi (výrobci displejů používají HD44780 nebo jeho ekvivalent). K napájení je potřeba 5V zdroj. Komunikovat lze po 4-bitové nebo 8-bitové paralelní sběrnici. Komunikační instrukce jsou popsány v dokumentu „LCD 1602A datasheet“, který je přílohou této práce. Řadič má v sobě uloženou základní znakovou sadu, ke které je možno definovat a uložit 8 uživatelských znaků, této možnosti využiji v následném vytváření vad pro otestování vyhodnocování navrženého testovacího nástroje.

Při vypracovávání práce jsem měl k dispozici 5 kusů toho displeje.

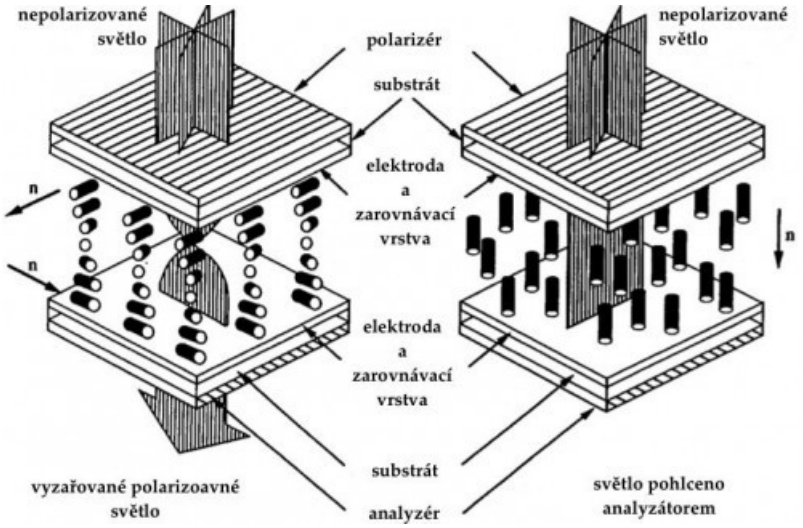
Obrázek 1: LCD Displej 1602A

Obrázek 2: Řadič LCD displeje

[7]

2.2. Technologie zobrazování LCD displeje 1602A

LCD displej je obvykle tvořen dvěma skleněnými deskami mezi, kterými se nachází aktivní zobrazovací prvky, pixely. Každý pixel je tvořen molekulami tekutých krystalů, uloženými mezi dvěma průhlednými vodivými pásky, elektrodami a dvěma polarizačními filtry, kdy osy polarizace filtrů jsou na sebe kolmé. Jako elektrody se používají průhledné vodivé pásky na jedné skleněné vrstvě a na ně kolmé pásky na druhé skleněné vrstvě. Pixel je tedy určen průsečíkem těchto dvou elektrod a je adresován sloupcem a řádkem. Jedná se o pasivní zobrazovací pole, a aby bylo dosaženo stálého obrazu, musí být snímek velmi rychle obnovován (tak aby to lidské oko nepostřehlo). Bez tekutých krystalů by světlo procházející jedním filtrem bylo zablokováno druhým. Při průchodu světla pixelem je polovina světla absorbována prvním polarizačním filtrem. Bez působení vnějšího elektrického pole jsou molekuly tekutého krystalu srovnány do spirálové struktury a otáčí polarizaci procházejícího světla o 90° což umožňuje průchod druhým polarizačním filtrem, pixel svítí (propouští světlo). V okamžiku vzniku elektrického pole mezi elektrodami jsou molekuly tekutého krystalu taženy rovnoběžně s polem, až nejsou stočené vůbec, čímž se zruší rotace vstupujícího světla a to je zablokováno druhým polarizačním filtrem, pixel nesvítí (nepropouští světlo). Princip je zobrazen na Obrázek 3.



Obrázek 3: Princip průchodu světla pixelem LCD

[12]

2.3. Připojení LCD displeje k PC

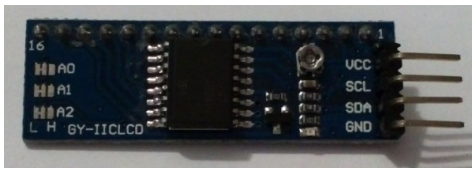
Vzhledem k tomu, že displej nepodporuje žádné standardní rozhraní používané v PC, musel jsem zvolit v hodný způsob pro jeho snadné připojení a napájení.

Poměrně rozšířeným způsobem připojení displejů k jednočipovým zařízením je převodník IICLCD, který umožňuje komunikaci, napájení a ovládání displeje po sběrnici I2C. Převodník obsahuje integrovaný obvod PCF8574, zajišťující převod zpráv I2C sběrnice na 8-bitový vstupně/výstupní paralelní port. Čtyři z těchto bitů jsou použity pro přenos dat do řadiče displeje, tři jsou použity pro řízení komunikace, zbylý bit ovládá přes tranzistor zapnutí podsvícení. Veškeré

napájení poskytuje I2C sběrnice. Pro ovládání kontrastu displeje je na převodníku umístěn potenciometr.

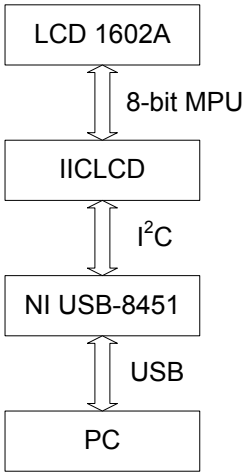
Jednou z variant jak komunikovat s převodníkem IICLCD z personálního počítače je použít převodník USB na I2C. Pro účely diplomové práce jsem použil převodník NI USB-8451, který tuto funkcionalitu zajišťuje. Převodník NI USB-8451 umožňuje komunikaci s I2C, SMBus a SPI zařízeními a je napájen z USB portu počítače.

Blokové schéma připojení LCD k PC je na Obrázek 6.



Obrázek 4: Převodník IICLCD

Obrázek 5: Převodník NI USB-8451



Obrázek 6: Blokové schéma připojení LCD k PC

[8], [15]

2.4. Použité PC a operační systém

Pro programování a testování software pro kontrolu LCD displeje jsem použil počítač (notebook HP Probook 4540s) vybavený dvoujádrovým procesorem Intel Core i5-3210M o frekvenci 2,5GHz (Turbo Frekvence 3,1GHz) a 8GB RAM. Pro připojení do sítě LAN je počítač vybaven síťovou kartou Realtek PCIe GBE (Gigabit Ethernet). Pro připojení dalších periférií jsou k dispozici čtyři USB porty (dva ve verzi 2.0 a dva ve verzi 3.0). Na počítači je nainstalován systém Microsoft Windows 7 Home Premium a Linuxová distribuce Fedora 20.

2.5. Kamera a kamerové vybavení

2.5.1. Kamera

Pro aplikace strojového vidění se nejčastěji používají průmyslové kamery, které se vyznačují těmito pro tento účel danými vlastnostmi, malými rozměry, robustní konstrukcí, odolností proti vibracím, prachu, změnám teploty apod.

Charakteristické parametry kamer:

- Rozlišení senzoru
- Velikost senzoru
- Barevná hloubka (barevná/černobílá)
- Typ senzoru (CCD/CMOS, řádkové/maticové)
- Doba expozice, rychlost
- Typ komunikačního rozhraní (analogové, digitální)
- Typ uchycení objektivu

Senzor

Základ každé kamery je senzor, který sloučí k převodu dopadajícího světla na vyjádření v elektronické podobě. Čip je tvořen jednotlivým pixely uspořádanými do matice (maticové kamery) nebo řádku (řádkové kamery). Počet pixelů na senzoru udává rozlišení kamery. Rozlišení se udává jako součin počtu bodů stran obrazu (1280 x 960) nebo celkovým počtem (1,3 Megapixel). Podle technologie převodu fotony na elektrický signál se senzory dělí na CCD a CMOS čipy.

Typ senzoru

Senzor CCD (Chanrge-Coupled Device) neboli zařízení s vázanými náboji, po expozici fotony je přenášen náboj vyvolaný dopadem mimo strukturu čipu, kde jsou obvody pro převod náboje na elektrické napětí a zesílení. Funkce pro zpracování náboje jsou umístěny mimo vlastní senzor. V případě potřeby je možné vyměnit externí elektroniku. Vyznačuje se vynikající kvalitou obrazu a vysokou flexibilitou. Nevýhodou je však vysoká cena, vysoká energetická náročnost, nízká snímkovací frekvence, omezená možnost miniaturizace a malá odolnost vůči přeexponování obrazu. Využívá se v aplikacích, kde je potřeba kvalitní obraz, jako je medicína, astronomie a další.

CMOS (Complementary Metal-Oxide-Semiconductor) senzor převádí náboj fotonu na napětí přímo v pixelu. Pro změnu vlastností čipu je potřeba celý čip vyměnit. Jeho výhodou je ucházející kvalita obrazu, nízká cena, vysoká snímkovací frekvence, nízká spotřeba, vysoká odolnost proti

přeexponování a velké možnosti miniaturizace. Využití nachází v aplikacích, s malým důrazem na vysokou kvalitu obrazu, především webkamery, mobilní telefony, bezpečnostní kamery a další.

Fyzická velikost senzoru

Velikost senzoru kamery se udává v palcích. Větší senzor, při použití se stejné optiky, poskytuje větší zorné pole, lepší obraz, menší šum a větší rozlišení.

Doba Expozice

Doba expozice udává, jak dlouho trvá naakumulování náboje v čipu při snímání jednoho snímku. Tento parametr je nastavitelný v rozsahu řádově jednotek sekund po jednotky nanosekund, tento rozsah se odvíjí od typu kamery. Převrácená hodnota expoziční doby udává rychlost snímkovací frekvence kamery ve snímcích za sekundu FPS (Frame Per Second). Snímání s dlouhou expoziční dobou lze snímat tmavé scény nebo scény s velkým zacloněním objektivu pro lepší hloubku ostrosti, na úkor snímkovací frekvence. S krátkou expoziční dobou lze snímat rychlé děje, vysokou snímkovací frekvencí, je však nutné silnější osvětlení.

Snímkovací frekvence a rozlišení souvisí s volbou komunikačního rozhraní, které zajišťuje přenos obrazu po standardizované sběrnici do zařízení sloužícího pro vyhodnocení.

Komunikační rozhraní

Kamera může disponovat celou řadu konektorů pro zprostředkování datového přenosu. Komunikační rozhraní kromě přenosu obrazových dat zajišťuje, přenos řídících a synchronizačních signálů, konfiguraci kamery, napájení kamery. Dnes nejpoužívanější digitální komunikační rozhraní pro kamery jsou:

- FireWire (rev .b), 400Mbit/s, 10m
- USB (2.0), 450Mbit/s, 10m
- USB (3.0) 5Gbit/s, 10m
- GigE (Gigabit Ethernet), 1Gbit/s, 100m
- Camera Link, Škálovatelné řádově stovky Mbit/s, 10m
- CoaXPress, Řádově desítky Gbit/s, 100m+

Zvolená kamera

Jako nástroj pro snímání obrazu za účelem optického testování LCD displeje jsem zvolil kameru BASLER ac2500-14gm snímající černobílý obraz viz. Obrázek 7. Kamera je vybavena CMOS senzorem o velikosti 1/2.5‘‘ s rozlišeními 5MPixelů (2592 x 1944). Komunikačním rozhraním kamery je GigE s možností napájení pomocí technologie POE (Power Over Ethernet). Pro napájení kamery jsem použil Ethernetový injektor modul.

Obrázek 7: Kamera s připevněným objektivem

[28], [9], [11]

2.5.2. Objektiv

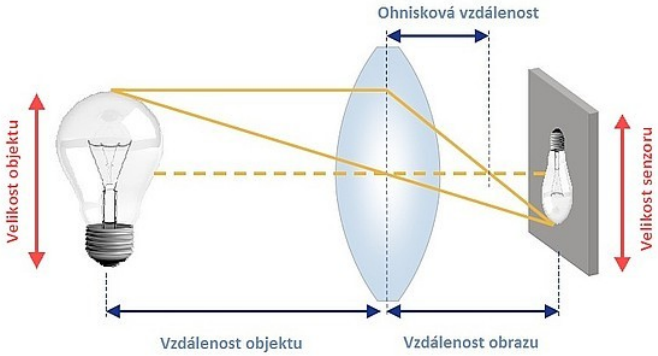
Pro získání kvalitního obrazu z kamery je potřeba zvolit vhodný objektiv, který zajistí koncentraci dopadajícího světla na senzor kamery tak, aby snímaný obraz byl dostatečně ostrý a měl odpovídající jas. Zaostření obrazu objektivem je možné zajistit buď manuálně, ručním otáčením zaostřovacím kroužkem, nebo automaticky (Autofokus), kdy je objektiv vybaven pohonem.

Geometrie zobrazení objektivem

Zobrazení předmětu objektivem je zobrazeno na Obrázek 8 a řídí se následující rovnicí pro návrh optické soustavy:

$$\frac{velikost\ objektu}{vzdálenost\ objektu} = \frac{velikost\ senzoru}{ohnisková\ vzdálenost}$$

Rovnice 1: Zobrazení objektivem



Obrázek 8: Zobrazení předmětu v kameře objektivem

Ohnisková vzdálenost objektivu

Ohnisková vzdálenost objektivu je údaj udávající vzdálenost čočky objektivu od senzoru kamery. Tím je dáno, jak velký předmět bude možné na senzoru kamery zobrazit. Ohnisková vzdálenost je buď pevně daná (výrobní řada např. 12, 16, 25, 35mm) nebo nastavitelné (fokusovatelné).

Vzdálenost objektu od objektivu

U běžných objektivů není maximální vzdálenost objektu od objektivu omezena, každý objektiv je schopen zaostřit na nekonečno. Konstrukce objektivu však udává minimální vzdálenost, na kterou je schopen zaostřit. Zkrácení vzdálenosti lze provést pomocí mezikroužků vkládaných mezi objektiv a kameru, za cenu zhoršení světelnosti objektivu.

Clona objektivu

Clona objektivu reguluje množství světla procházející objektivem. Funguje stejně jako zornička v lidském oku. Clonu tvoří dle potřeby nastavitelný kruhový otvor uprostřed objektivu. Konstrukce je většinou lamelová viz Obrázek 9. Clony se odlišují různou velikostí, konstrukcí, tvarem a materiálem. Pokud je clona otevřena naplno, odpovídá to jejímu maximálnímu průměru a to udává minimální clonové číslo. Při plném uzavření udává maximální clonové číslo. Její nastavení umožňuje v kombinaci s rychlostí závěrky nastavit správné množství světla pro danou expozici. Clona také ovlivňuje hloubku ostrosti snímaného obrazu. Za horších světelných podmínek lze pořídit ostrý snímek při otevření clony (menší clonové číslo). Za dobrých světelných podmínek při otevření clony a snížením expozičního času se sníží hloubka ostrosti (oddělení objektu od pozadí). Přivření clony za dobrých světelných podmínek hloubku ostrosti zvyšuje.

Obrázek 9: Lamelová clona objektivu

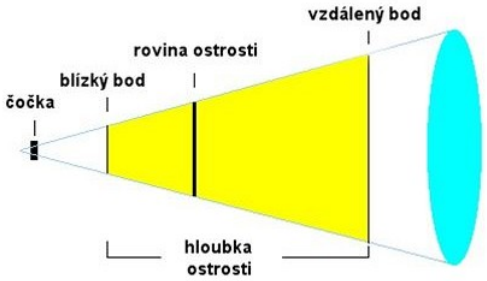
Světelnost objektivu

Světelnost objektivu udává schopnost objektivu propouštět světlo. Hodnota světelnosti je bezrozměrné číslo, které udává poměr mezi osvětlením senzoru a jasu snímaného objektu (předmět v nekonečnu a na optické ose objektivu). Propustnost objektivu by byla rovna jedné (100%), pokud všechno světlo projde objektivem, až na senzor kamery. V praxi, je ale vždy část světla pohlcena objektivem a čím více čoček objektiv obsahuje, tím více světla pohlcuje. Světelnost je přímo úměrná propustnosti světla a nepřímo závisí na druhé mocnině indexu lomu skla čoček objektivu a ohniskové vzdálenosti. S vzrůstajícím indexem lomu světla a ohniskovou vzdáleností se světelnost snižuje. Základní řada clonových čísel je odvozena od násobku clonového čísla 1 s odmocninou ze dvou (1; 1,4; 2,0; 2,8; 4,0; 5,6; 8; atd.). Často se však pojem světelnost nahrazuje minimální hodnotou clonového čísla objektivu a toto číslo tvoří jednu ze základních charakteristik objektivu. Označuje se písmenem F společně s číslem, např. F/2,8.

Ostření objektivu a hloubka ostrosti

Aby se snímaný objekt jevil ostrý, musíme zvolit adekvátní rovinu zaostření. Rovina zaostření je rovina rovnoběžná s rovinou senzoru kamery a kolmá k ose objektivu. Body v této rovině se na senzoru zobrazí ostře a body mimo budou rozostřené. Rovina prochází zaostřeným předmětem. Ostření je provedeno mechanickým pohybem jedné či více čoček vůči sobě nebo senzoru kamery. Tato operace mění ohniskovou vzdálenost objektivu.

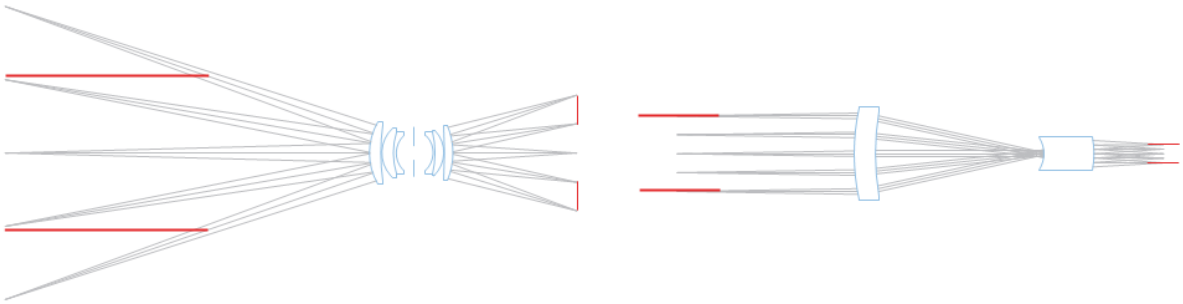
Předmět či jeho části obvykle neleží v dané přesné vzdálenosti od objektivu a to způsobuje malé neostrosti, které jsou okem nepostřehnutelné. Tento rozsah se nazývá hloubka ostrosti a udává rozdíl vzdálenosti nejvzdálenějšího od nejbližšího objektu, viz Obrázek 10, jevící se lidskému oku stále ostře. Větší ohnisková vzdálenost snižuje hloubku ostrosti objektivu. S větším zacloněním objektivu hloubka ostrosti roste. S rostoucí vzdálenosti objektu od objektivu roste i hloubka ostrosti



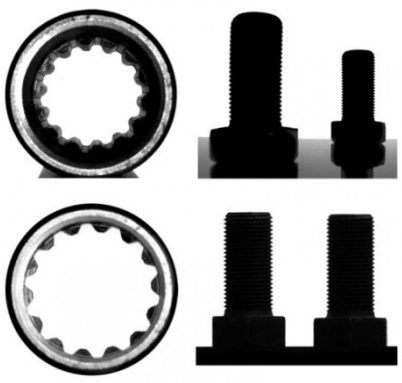
Obrázek 10: Hloubka ostrosti

Telecentrické objektivy

Pro velmi přesné měření rozměrů a tvarů v aplikacích strojového vidění se často používá speciální typ objektivů, tzv. telecentrické objektivy. Tyto objektivy jsou schopny zobrazovat předměty s konstantním zvětšením nezávisle na vzdálenosti od objektivu (eliminace perspektivy). Srovnání zobrazení mezi klasickým a telecentrickým objektivem je na Obrázek 11 a Obrázek 12.



Obrázek 11: Srovnání zobrazení (nalevo) klasickým a (napravo) telecentrickým objektivem



Obrázek 12: Zobrazení objektu (nahoře) klasickým a (dole) telecentrickým objektivem

Obrázek 12 ukazuje rozdíl mezi použitím klasického (nahoře) a telecentrického (dole) objektivu v praxi. Na válcovíté součástce (nalevo) lze vidět, že při zobrazení klasickým objektivem jde vidět její vnitřní stěna, která znesnadňuje měření součástky. U zobrazení dvou identických šroubů (napravo) umístěných v různé vzdálenosti od sebe lze vidět značný rozdíl v jejich zobrazení klasickým objektivem. Telecentrický objektiv odstraňuje perspektivu a tyto jevy eliminuje.

Zvolený objektiv

Jako objektiv, pro výše zvolenou kameru, jsem zvolil klasický manuálně ostřitelný objektiv Kowa LM25JC10M 2/3‘‘ s ohniskovou vzdáleností 25mm a světelností F1,8 viz Obrázek 7.

[19], [24], [29]

3. Vývojový software

Algoritmus pro zpracování obrazové informace jsem naprogramoval ve dvou vývojových prostředích: LabVIEW a C++. Podrobnosti jsou uvedeny v následujících kapitolách.

3.1. NI LabVIEW

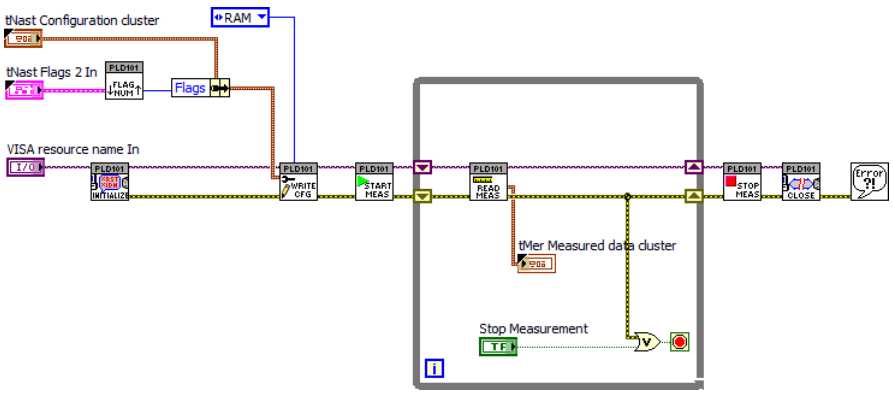
Pro vývoj testovacího nástroje monochromatických LCD displejů je použito vývojové prostředí LabVIEW od společnosti National Instruments. Jedná se o komplexní vývojové prostředí rozšiřitelné o množství modulů umožňující sběr, analýzu a vizualizaci dat. Toto vývojové prostředí jsem měl k dispozici ve verzi 2013 a studentské licenci.

LabVIEW nabízí grafické programování jehož základním stavebním prvkem je SW entita nazývaná virtuální přístroj „Virtual Instrument“ (dále jen VI), který svým chováním představuje skutečný přístroj. VI se skládá ze dvou základních částí: čelního panelu „Front panel“ a blokového diagramu „Block diagram“:

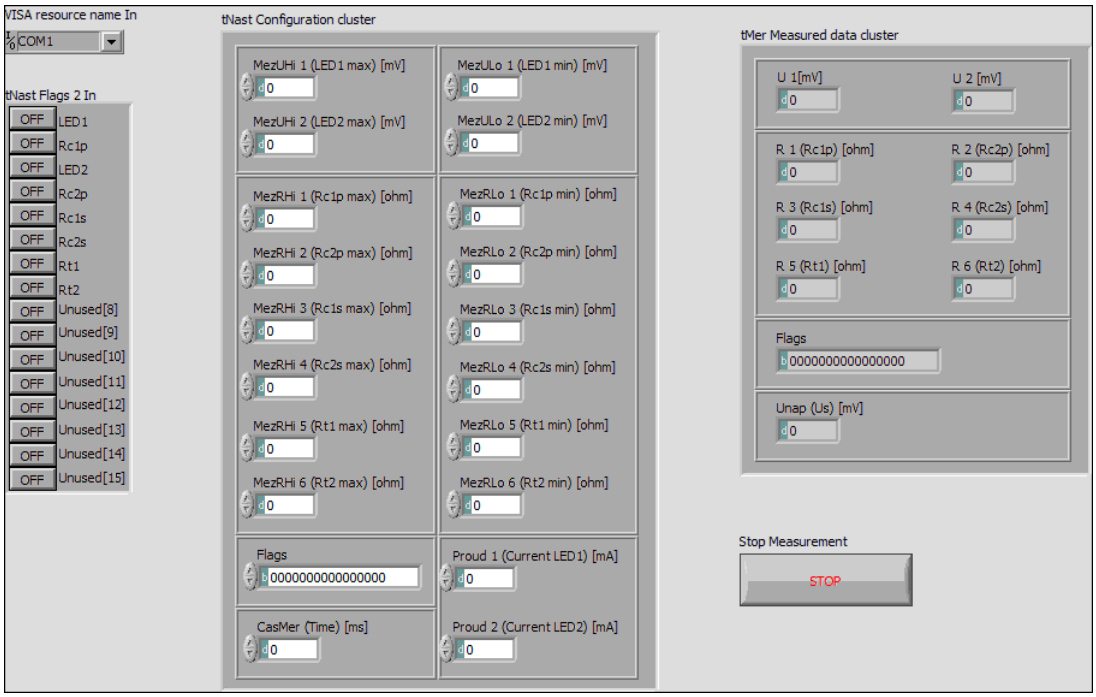
- **Čelní panel „Front panel“** je grafickým uživatelským rozhraním, které představuje čelní panel fyzického přístroje. Obsahuje prvky pro ovládání a indikaci (tlačítka, LED indikátory, grafy, textové pole a celou řadu dalších prvků). Čelní panel se ovládá myší nebo klávesnicí.
- **Blokový diagram „Block diagram“** představuje činnost virtuálního přístroje, která je definována jeho vnitřními datovými toky a volanými funkcemi. Blokové schéma je tvořeno ikonami reprezentujícími v koncových bodech ovládací a indikační prvky čelního panelu a ve svých uzlových blocích zpracovávající se data. Blokový diagram je zdrojovou podobou každé aplikace.

VI má hierarchickou a modulární strukturu, jediné VI může v krajním případě představovat celý program nebo plní funkci podprogramu a pak se nazývá „subVI“. Každé VI je pro styk s dalšími VI reprezentováno ikonou s konektory pro připojení vstupních a výstupních signálů na jeho ovládací a indikační prvky. Ikona se poté zobrazuje v blokovém diagramu při použití jako subVI.

V práci uvedené ukázky blokových diagramů z vývojového prostředí LabVIEW nejsou kompletním zdrojovým kódem, protože některé prvky jsou více úrovňové, například rozhodovací struktury, kdy je vidět pouze její část.



Obrázek 13: Ukázka blokového diagramu VI



Obrázek 14: Ukázka čelního panelu VI

Obrázek 15: Logo LabVIEW

[14]

3.2. Podpůrný SW pro zpracování obrazu v NI LabVIEW

Zpracování obrazu ve vývojovém prostředí NI LabVIEW zajišťují následující dodatečně instalované SW moduly. Vision Acquision Software, Vision Development Module a Vision Asistant, které budou dále popsány.

3.2.1. Vision Acquision Software

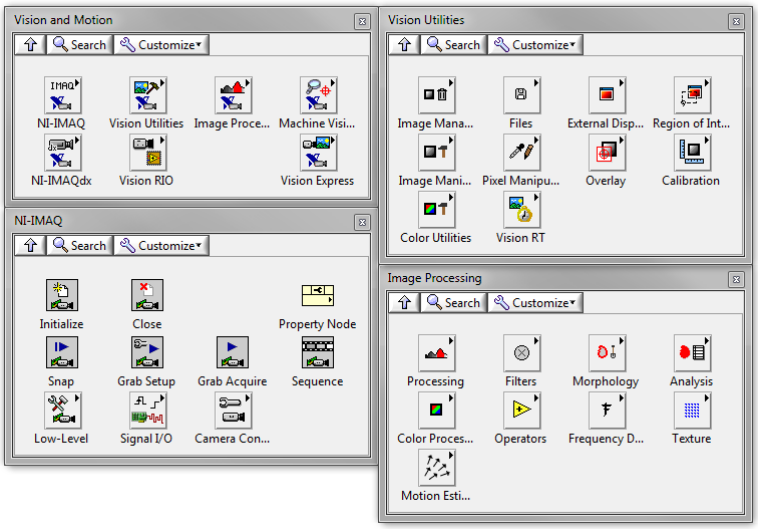
Vision Acquision Software od National Instruments je softwarový ovladač pro získávání, zobrazování, ukládání a monitorování obrazových informací z řady typů kamer. Obsahuje nástroje pro konfiguraci nastavitelných parametrů kamer. Vytváří univerzální programové rozhraní pro komunikaci s kamerami připojených přes nejrůznější komunikační rozhraní, jako je Gigabit Ethernet, FireWire, USB 3.0, CameraLink, a další. Je plně kompaktibilní s LabVIEW, C, C++, C#, Visual Basic, and Visual Basic .NET.

[16]

3.2.2. Vision Development Module

Vision Development Module National Instruments je softwarový modul, který je navržen pro vývoj aplikací strojového vidění a zpracování obrazu. Obsahuje pokročilé funkce pro detekci přítomnosti objektů, zaměření jejich pozice, identifikace objektů a jejich měření.

Oba softwarové moduly podporují grafické vývojové prostředí NI LabVIEW a také C, C++, C# a Visual Basic .NET pro operační systém Windows. Na Obrázek 16 uvádím ukázku některých dostupných funkcí pro strojové vidění v LabVIEW.



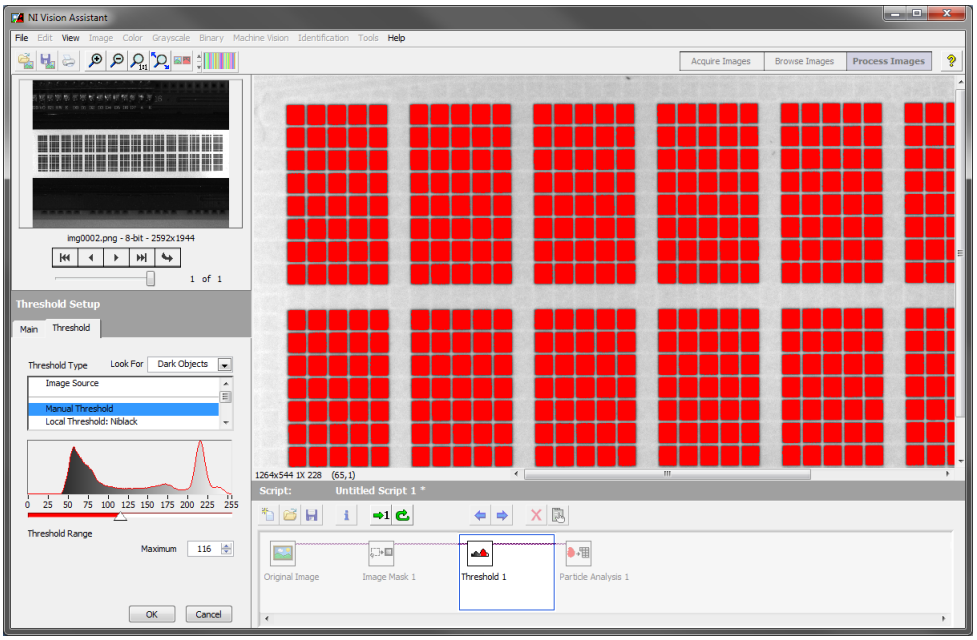
Obrázek 16: Ukázka funkcí pro strojové vidění v LabVIEW

[18]

3.2.3. Vision Assistant

Součástí softwarového modulu Vision Development Modul je nástroj Vision Assistant. Jedná se o vývojový nástroj, který umožňuje metodou krok-po-kroku navrhovat aplikace strojového vidění. Při návrhu algoritmu vývojáři umožňuje u každého kroku zpracování obrazu nastavit parametry použité operace zpracování obrazu a ihned mu umožňuje vidět výsledek. Navržený algoritmus je následně možno vygenerovat jako VI v LabVIEW, kód v C++, .NET, nebo uložit jako skript pro další použití. Nástroj umožňuje zpracování obrazu přímo z kamery, nebo z obrazových souborů uložených na disku.

Na Obrázek 17 je ukázka z vývoje vyhodnocovacího algoritmu aplikace pro testování LCD displejů.



Obrázek 17: Ukázka návrhu algoritmu strojového vidění v NI Vision Assistant

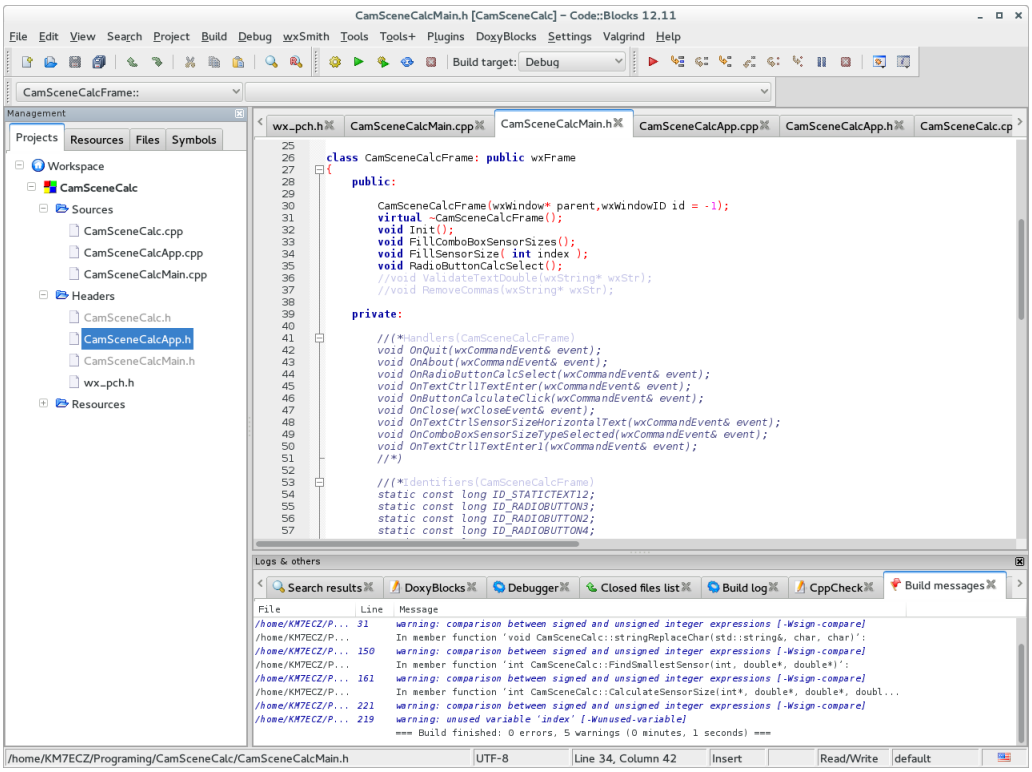
[17]

3.3. Nástroje pro programování a zpracování obrazu v jazyku C++

Pro porovnání nástroje pro kontrolu monochromatických LCD displejů navrženého v LabVIEW a naprogramování nástroje pro výpočet pracovní vzdálenosti objektivu, jsem zvolil multiparadigmatický programovací jazyk C++. Zvolené nástroje pro programování v C++ a podrobnosti jsou popsány v následujících kapitolách.

3.3.1. Code::Blocks

Jako vývojové prostředí pro programování v C++ jsem zvolil vývojové prostředí Code::Blocks ve verzi 12.11-2. Jedná se o multiplatformní vývojové prostředí pro programovací jazyky C, C++, Fortran, které má konzistentní vzhled na různých platformách. Code::Blocks je navrženo s ohledem na snadnou rozšiřitelnost zásuvnými moduly tzv. „pluginy“. Zásuvné moduly poskytují podporu různých překladačů „compilers“ a nástrojů pro hledání chyb v kódu tzv, „debuggers“. Na Obrázek 18 je vidět ukázka z vývojového prostředí Code::Blocks.



Obrázek 18: Vývojové prostředí Code::Blocks

Obrázek 19: Logo Code::Blocks

[2]

3.3.2. GNU GCC

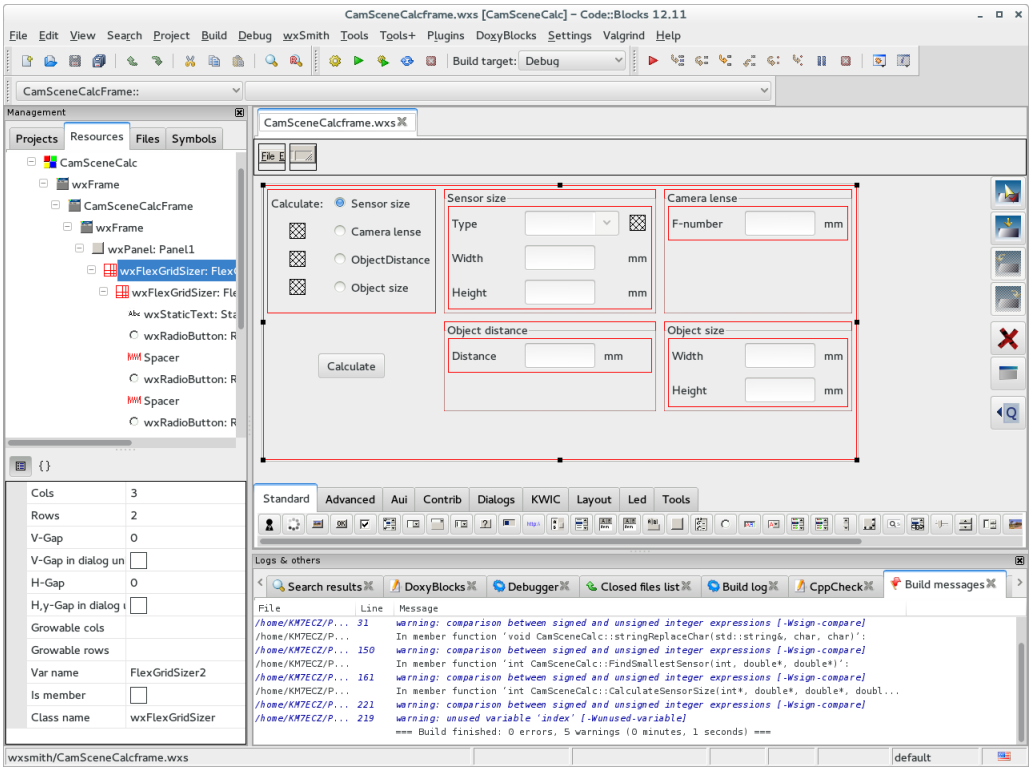
Jako kompilátor jsem v Code::Blocks zvolil GCC (GNU Compiler Collection) překladač ve verzi 4.8.2-7. GCC je standardním překladačem v operačních systémech „Open Source“ unixového typu, ale využívá se i v některých komerčních systémech jako je Mac OS X. Pro operační systém Microsoft Windows existují portace např. MinGW. Kompilátor obsahuje podporu pro překlad jazyku C/C++, Fortran, Java a dalších.

Obrázek 20: Logo CNU GCC

[6]

3.3.3. wxWidgets

Pro snadnější návrh GUI (Grafického Uživatelského Rozhraní) v jazyce C++ jsem ve vývojovém prostředí Code::Blocks využil modul wxSmith, který využívá multiplatformní knihovnu wxWidgets ve verzi 2.8.12-9. wxSmith Je grafický nástroj pro snadnější návrh uživatelských rozhraní ve vývojovém prostředí Code::Blocks. wxWidgets je C++ knihovna pro návrh aplikací pro Windows, Mac OS X a Linux. Obsahuje také podporu pro programovací jazyky Python, Perl, Ruby, C# a mnoho dalších. Knihovna wxWidgets poskytuje aplikacím skutečný přirozený vzhled. Tyto nástroje jsou zdarma a získal jsem je stažením z oficiálního repozitáře operačního systému Fedora 20.



Obrázek 21: Vývojové prostředí Code::Blocks s grafickým modulem wxSmith

Obrázek 22: Logo wxWidgets

[31], [32], [33]

3.3.4. OpenCv

Opencv (Open Source Computer Vision) je multiplatformní knihovna algoritmů pro strojové vidění vydaná pod BSD licencí a je jí možné volně využít v akademické i komerční sféře. Knihovna je napsaná a optimalizovaná v jazyce C/C++. Umožňuje využívat zpracování na více jádrech procesoru a zaměřuje se na zpracování obrazu v reálném čase. Podporuje technologii CUDA a OpenCL, umí tedy využít hardwarové akcelerace na různorodých výpočetních platformách. Má rozhraní pro využití v programovacích jazycích C, C++, Java, Python a je podporována v operačních systémech Windows,

Linux, iOS a Android. Jedná se tedy o oblíbenou, rozšířenou po celém světě, široce dokumentovanou knihovnu umožňující kromě návrhu a ladění algoritmů zpracování obrazu také možnost vizualizace zpracovávaných obrazových signálů. Využívá se v nejrůznějších odvětvích od interaktivního umění, přes důlní inspekci, skládání map, či tisíců dalších využití v robotice. OpenCV má komunitu čítající více jak 47 tisíc lidí.

Pro návrh testovacího nástroje monochromatických LCD displejů jsem použil OpenCv ve verzi 2.4.7 dostupnéhou z repozitáře operačního systému Fedora 20.

Obrázek 23: Logo OpenCv

[20], [21], [22]

4. Nástroj pro výpočet pracovní vzdálenosti objektivu

Při návrhu scény pro strojové vidění je vždy potřeba dopočítat jeden údaj z rovnice geometrie zobrazení objektivem uvedené v kapitole 2.5.2, proto vznikl dále popsaný tento nástroj pro jeho výpočet.

Ve vývojovém prostředí Code::Blocks v programovacím jazyce C++ jsem naimplementoval aplikaci pro výpočet pracovní vzdálenosti objektivu od snímaného objektu na základě parametrů kamery a objektivu. Nástroj jsem nazval CamSceneCalc (Camera Scene Calculator). Nástroj jsem naprogramoval v programovacím jazyce C++ v prostředí Code::Blocks grafickou knihovnou wxWidgets na operačním systému Fedora.

Nástroj jsem navrhnul tak, aby bylo možné vypočítat nejen vzdálenost snímaného objektu od objektivu „Object Distance“, ale i ostatní parametry snímací scény jako je velikost senzoru kamery „Sensor size“, ohniskovou vzdálenost objektivu „Camera lense Fnumber“ a velikost snímaného objektu „Object size“. Grafický vzhled nástroje je uveden na Obrázek 24. Výpočet parametrů snímací scény se řídí vztahem:

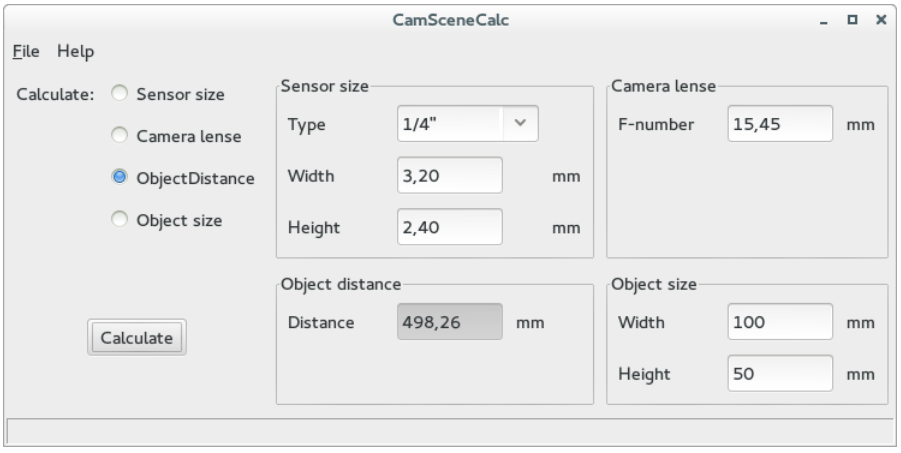
$$ohnisková\ vzdálenost = \frac{velikost\ senzoru\ \times vzdálenost\ objektu}{velikost\ objektu + velikost\ senzoru}$$

Rovnice 2: Výpočet ohniskové vzdálenosti objektivu

Aby uživatel nástroje nemusel znát přesnou velikost senzoru kamery, má nástroj databázi základních typů senzorů, tuto databázi tvoří textový soubor, do kterého lze snadno libovolným textovým editorem přidat další typy senzorů a jejich odpovídající rozměry.

Při výpočtu hodnoty ohniskové vzdálenosti objektivu a vzdálenosti snímaného objektu od objektivu je brán zřetel, aby se snímaný objekt zobrazil na senzoru kamery celý. Při výpočtu velikosti senzoru je vypočítána jeho potřebná velikost a vybrán nejbližší vhodný typ senzoru.

Zdrojový kód aplikace a spustitelný soubor pro Linux se nachází na CD přiloženém k této práci.



Obrázek 24: Nástroj pro výpočet pracovní vzdálenosti objektivu

[2], [3], [6], [19], [31], [32]

5. Snímací scéna a testovací snímky

Následující kapitoly popisují vytvoření snímací scény testovaného displeje a nasnímání testovacích snímků pro pozdější analýzu s pomocí SW nástroje pro ovládání a komunikaci s displejem.

5.1. Vytvoření snímací scény

Před návrhem a implementací samotného vyhodnocovacího algoritmu je potřeba vytvořit snímací scénu a nasnímat testovací snímky. Pro uchycení jsem použil stativ přípravek pro flexibilní nastavení a uchycení kamery s objektivem kolmo nad snímanou oblast. Do snímací oblasti jsem umístil displej vložený do nepájivého pole, který je přes převodníky připojen k PC. Protože displeje, které jsem měl k dispozici, byly vybaveny LED diodovým podsvícením, rozhodl jsem se ho využít a nahradit jím nasvícení scény. Použití nasvícení by bylo problematické hlavně z důvodu lesklé povrchové úpravě ochranného skla LCD. Bylo by nutné použít nasvícení pod úhlem, který by nezpůsoboval odlesky na sklíčku displeje. Nasvícení pod takovým úhlem, ale způsobuje stín za uzavřenými pixely LCD displeje. Tento stín by poté nepříznivě ovlivňoval vyhodnocování a muselo by se přistoupit k osvícení displeje z minimálně dvou zdrojů světla.

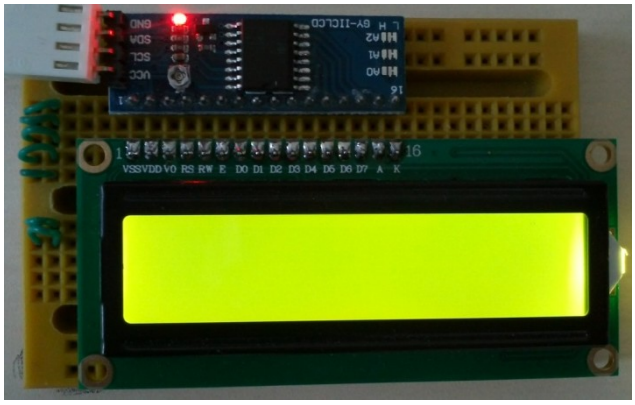
Na Obrázek 25 je zobrazena výsledná snímací scéna s kontrolovaným LCD displejem. Výsledný snímek pořízený kamerou je na Obrázek 26.

Obrázek 25: Snímací scéna

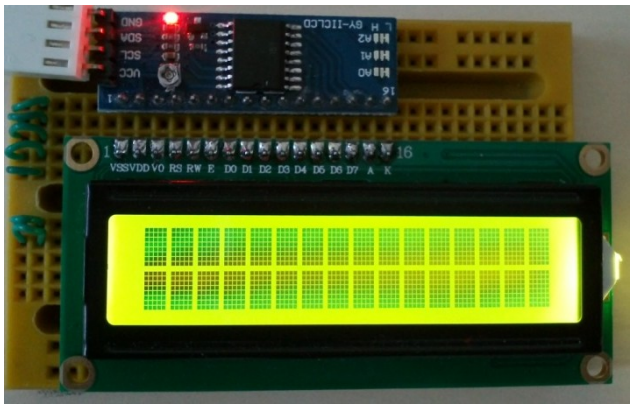
Obrázek 26: Snímek z kamery pro displej bez zobrazených znaků

5.2. Komunikace s displejem

Aby bylo možné kontrolovat funkčnost LCD displeje, bylo potřeba nejprve na něm zobrazit vhodnou informaci a rozsvítit podsvícení. Jako vhodné se ukázalo využít plného znaku obsaženého ve znakové sadě řadiče a tento znak zobrazit na všech dostupných pozicích displeje. Tento stav ověří funkčnost všech zobrazovacích segmentů, pixelů, displeje.



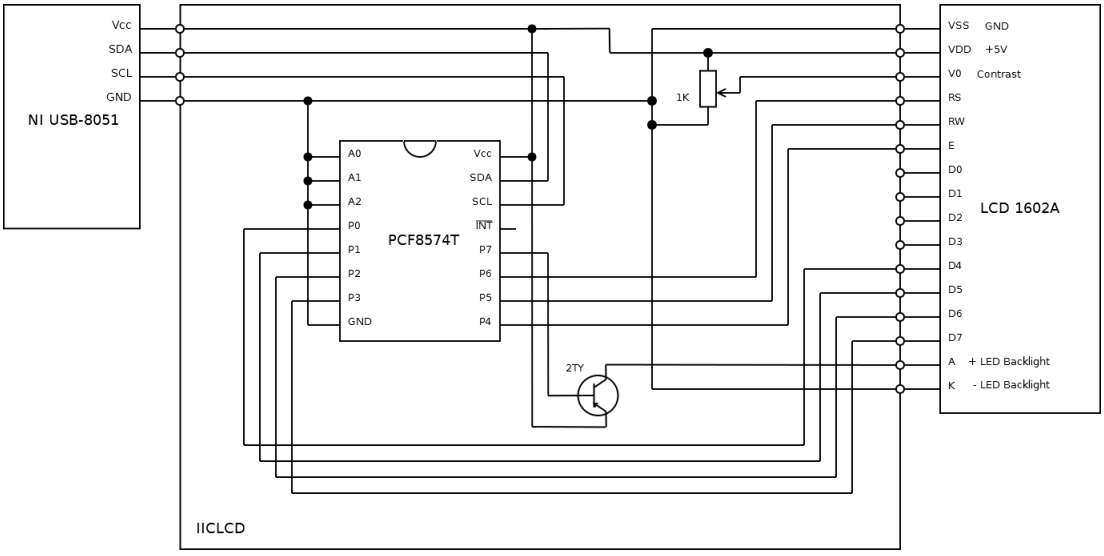
Obrázek 27: LCD displej se zapnutým podsvícením



Obrázek 28: LCD displej zobrazující plné znaky na všech pozicích

Pro účel komunikace s LCD displejem jsem v LabVIEW naprogramoval aplikaci využívající funkcí pro komunikaci po I2C sběrnici nainstalovaných spolu s driverem pro převodník NI USB-8451.

Protože displej standardně komunikuje po 8-bitové paralelní sběrnici s třemi bity (E, RW, RS) řídících přenos a převodník IICLCD poskytuje pouze 8-bitů je potřeba řadič displeje přepnout do režimu komunikace po 4-bitech. Poté je možno zprávu rozdělit a poslat do displeje postupně, nejdřív nejvýznamnější bity a poté méně významné bity. Řadič displeje si již tuto zprávu složí. Schéma zapojení je na Obrázek 29.



Obrázek 29: Schéma propojení displeje s převodníky

Základem vytvořeného nástroje je samostatné VI (I2CLCD_Server) viz. Obrázek 32, které umožňuje na základě vstupu „Action“ provést tyto typy operací: Inicializaci, vyčištění zobrazovaných informací na displeji, zobrazení definovaného textu, uložení uživatelského znaku a uzavření spojení.

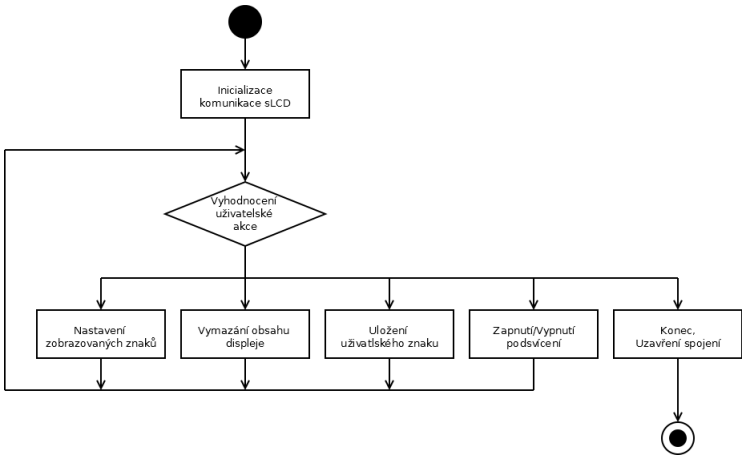
Čelní panel SW nástroje obsahuje ovládací prvky pro ovládání displeje viz Obrázek 33. Tlačítkem „Backlight LCD“ lze vypnout nebo zapnout podsvícení displeje. Tlačítko „Set LCD Text“ umožňuje na displeji zobrazit libovolný znak uložený v řadiči LCD displeje definovaný jeho adresou viz Obrázek 30. Do paměti řadiče (CG RAM) lze tlačítkem „Write Pattern LCD“ a adresou „Pattern

Location“ uložit 8 uživatelsky definovaných znaků (první sloupec v tabulce základní znakové sady LCD), jeho vzhled je možné definovat ovládacím prvkem „Pattern“. V případě potřeby je možné na displej vypsat klasický text tlačítkem „Set Normal Text LCD“.

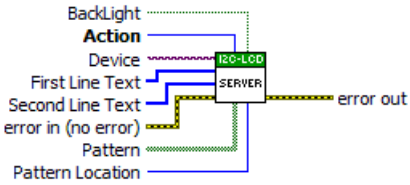
Obrázek 30: Základní znaková sada LCD displeje

SW nástroj pro komunikaci tvoří dvě smyčky viz Obrázek 34, horní smyčka slouží pro zachycení uživatelských událostí na GUI, jako je stisknutí tlačítka na čelním panelu a dolní, která má v sobě výkonný kód. Obě smyčky jsou synchronizovány frontou, do které horní smyčka vloží požadavek a spodní na základě druhu požadavku jej vykoná. Vývojový diagram běhu aplikace IICLCD

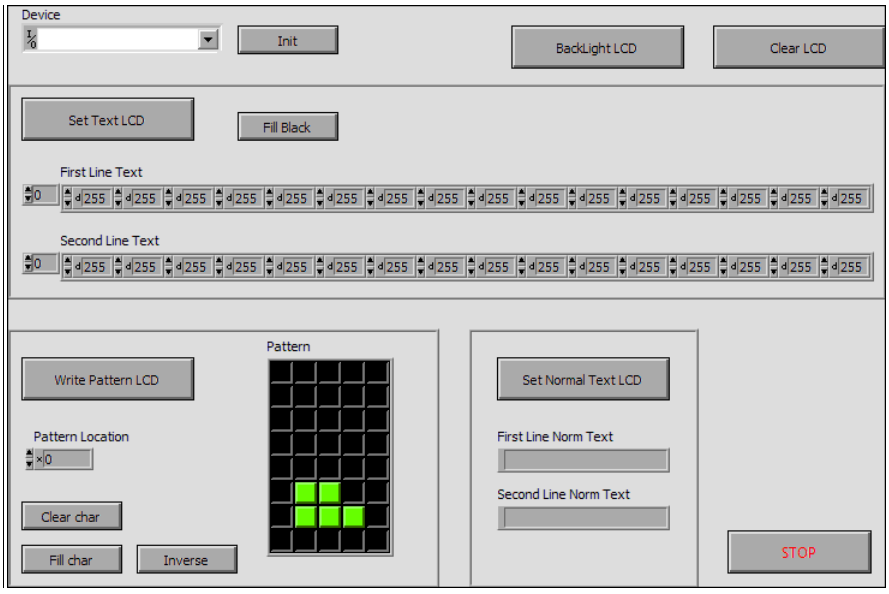
Naprogramovaný ovladač dále použiji jako subVI v návrhu samotného testovacího nástroje.



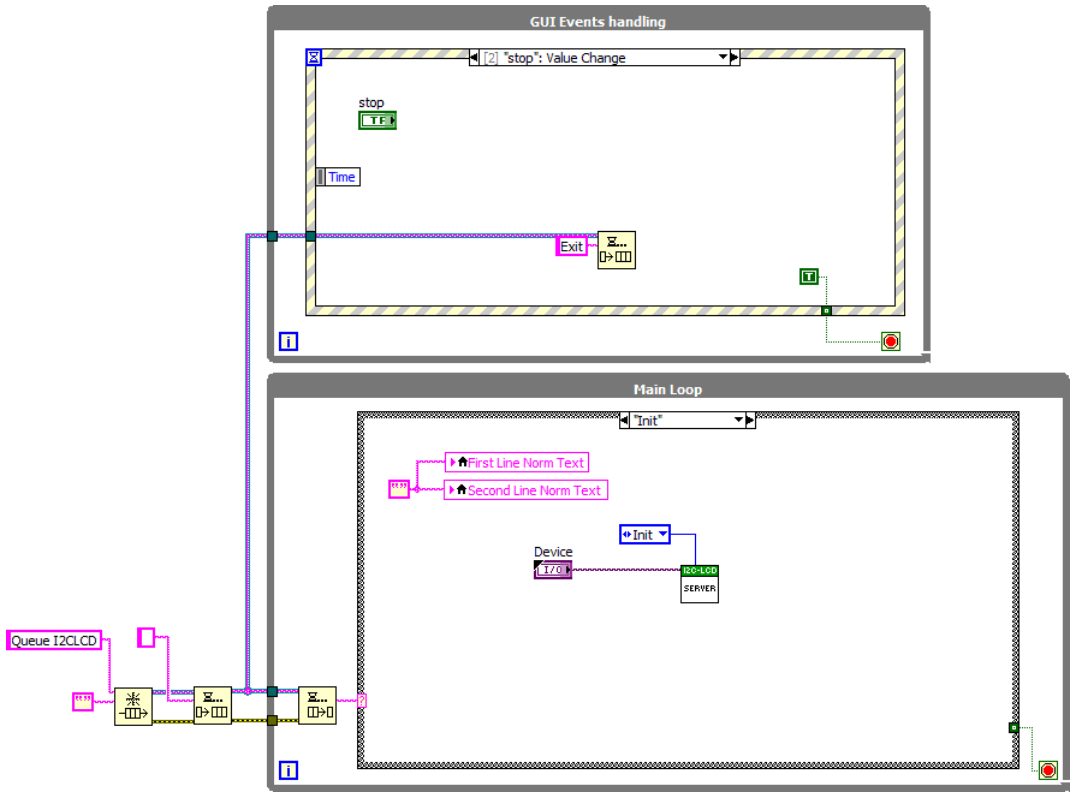
Obrázek 31: Vývojový diagram aplikace pro komunikaci s LCD displejem



Obrázek 32: VI I2CLCD Server



Obrázek 33: Čelní panel Aplikace pro komunikaci s LCD displejem



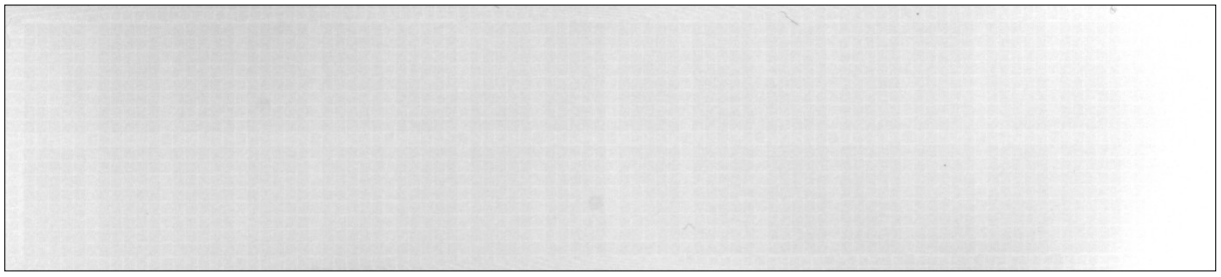
Obrázek 34: Blokový diagram aplikace I2CLCD

[1], [8], [7], [10], [15], [23], [30]

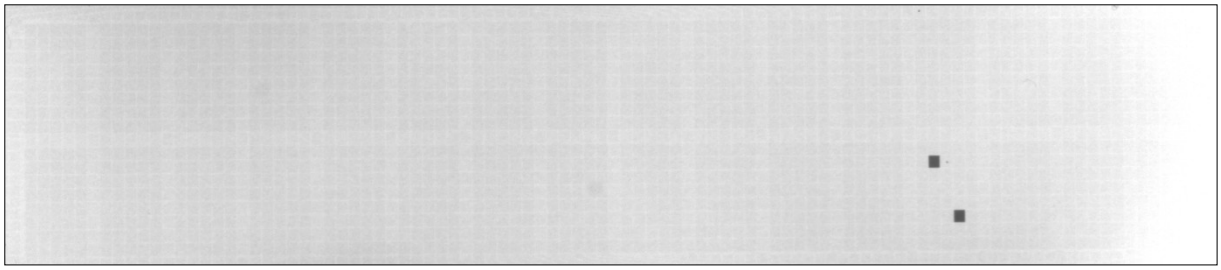
5.3. Nasnímání Testovacích snímků

Pro účely návrhu vyhodnocovacího algoritmu jsem nasnímal snímky reprezentující správné zobrazení displeje a snímky s uměle vytvořenou vadou na displeji. Vadu jsem uměle vytvořil vlastním definovaným znakem nebo mechanickým poškozením displeje.

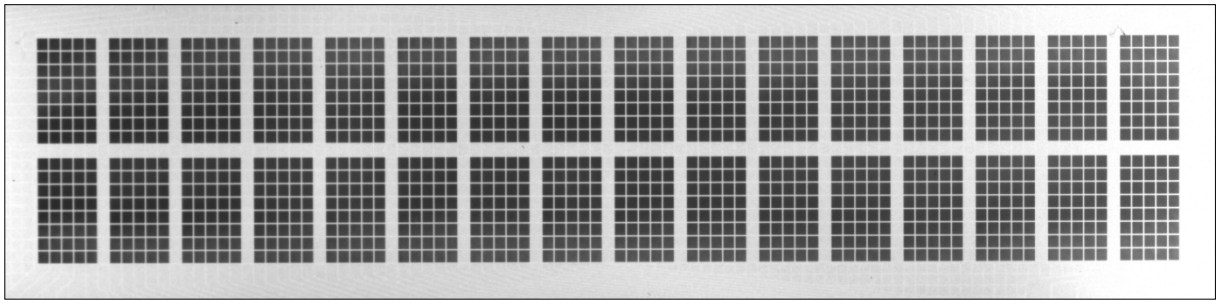
Následující snímky jsou typickými představiteli všech testovacích snímků. Jsou oříznuty na zájmovou oblast. Další testovací snímky jsou umístěny na přiloženém CD.



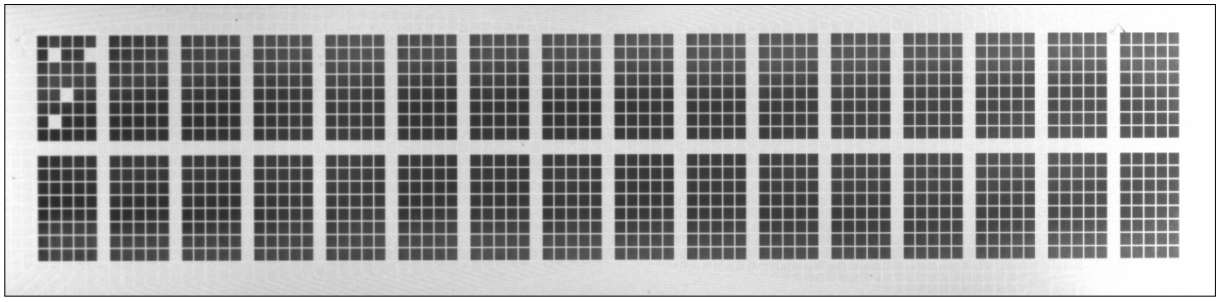
Obrázek 35: Podsvícený LCD pro analýzu pozadí



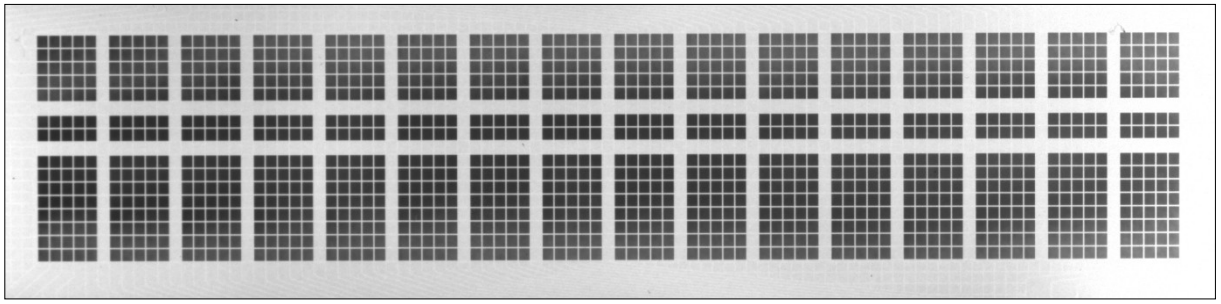
Obrázek 36: LCD bez znaků s vadnými pixely



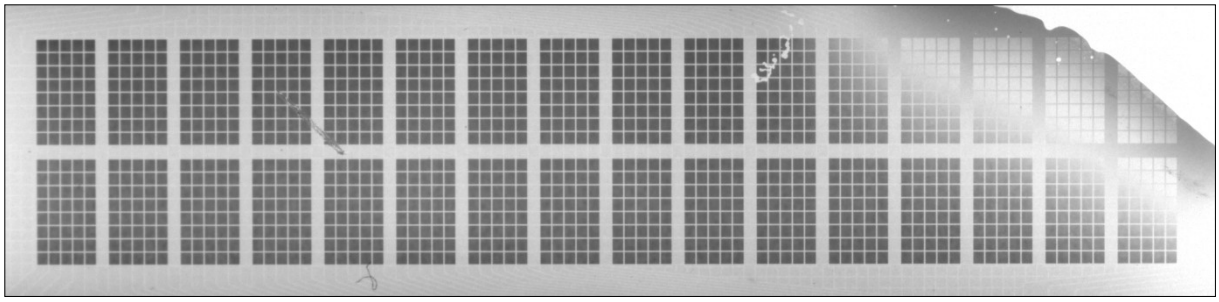
Obrázek 37: LCD bez vad zobrazení



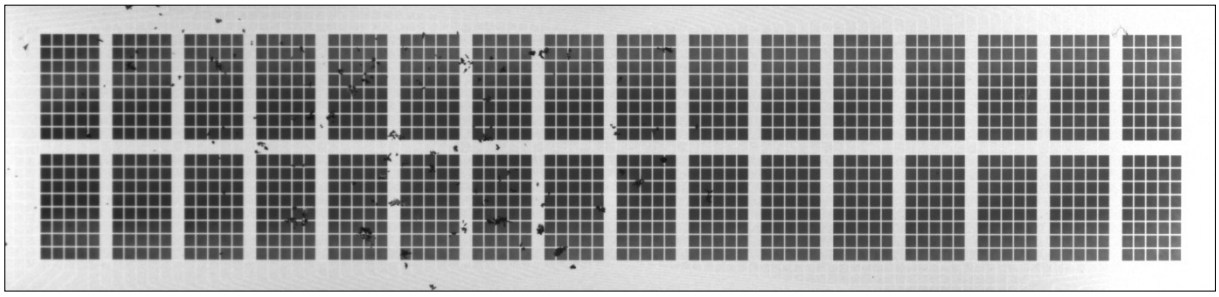
Obrázek 38: LCD s vadnými pixely



Obrázek 39: LCD s vadným celým řádkem pixelů



Obrázek 40: Mechanicky poškozený LCD



Obrázek 41: LCD s hrubými nečistotami

6. LCD Tester

Následující kapitoly popisují návrh a implementaci algoritmu pro kontrolu LCD displejů a vyhodnocovací aplikace v jednotlivých vývojových prostředích, která jsou zmíněna v kapitole 3. Nejprve jsem navrhnul a naimplementoval nástroj ve vývojovém prostředí LabVIEW a následně vyhodnocovací algoritmy v jazyce C++, pro porovnání časové náročnosti.

6.1. Předpoklady pro správnou funkčnost vyhodnocovacího algoritmu

Předpokladem pro správnou funkčnost vyhodnocovacího algoritmu je skutečnost, že displej bude do snímací scény umístěn vždy stejně orientován vzhledem orientaci kamery a na stejnou pozici. Snímací scéna musí být zatemněna, aby vyhodnocení neovlivňovalo vnější parazitní světlo. Objektiv kamery musí být správně zaostřen.

6.2. Kontrolované vlastnosti vyhodnocovacím algoritmem

Vyhodnocení správnosti zobrazení LCD displeje musí zohledňovat několik skutečností správného zobrazení.

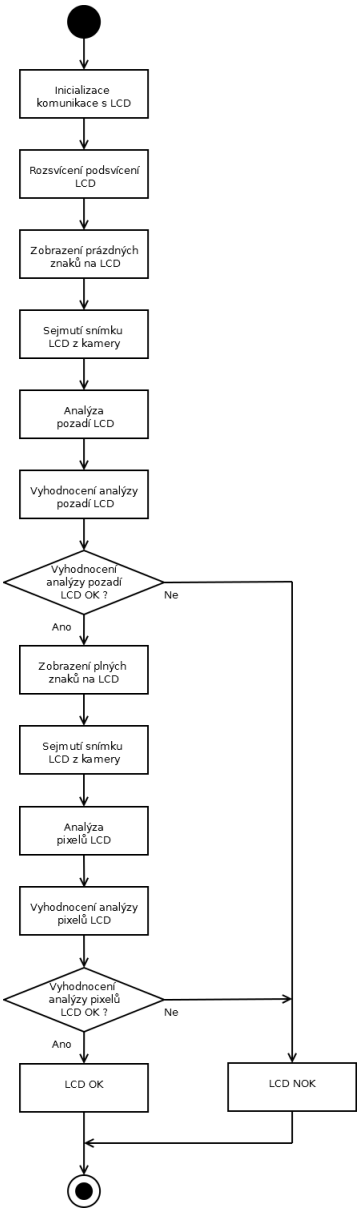
Zcela prvním kontrolovaným parametrem LCD displeje je, že funguje podsvícení a sklo LCD displeje a pozadí neobsahuje žádné aspekty mechanického poškození, jako jsou praskliny, škrábance či nečistoty. Tento stav se nejlépe kontroluje na rozsvíceném displeji, který nezobrazuje žádné znaky. Pokud tato kontrola vyhodnotí displej jako poškozený, žádné další testy již provedeny nebudou a kontrola je ukončena.

Pokud předešlá kontrola vyhodnotí displej bez závady, pak dalším hlavním aspektem bezchybnosti displeje je správná funkce všech pixelů, kdy u displeje 1602A tedy pixely nepropouštějí světlo. Pokud některý z pixelů nefunguje, považuje se to za vadu, protože u monochromatických LCD displejů může způsobit nečitelnost zobrazovaného znaku.

Další zkoumanou vlastností při kontrole je správná velikost a tvar jednotlivých pixelů. Pokud bude velikost pixelu menší nebo větší než uživatelsky definovaná odchylka pak je pixel považován za vadný.

Posledním kontrolovaným aspektem je množství světla propuštěného pixelem, pixel by neměl v ideálním případě propouštět žádné světlo, ale v reálných podmínkách v závislosti na technologii LCD displeje a síly podsvícení dochází k propouštění části světla.

Na Obrázek 42 je znázorněn vývojový diagram průběh kontroly vyhodnocovacím algoritmem, který jsem rozdělil do dvou hlavních částí, analýza pozadí a analýza pixelů.



Obrázek 42: Vývojový diagram kontroly LCD

6.3. Referenční korektní hodnoty pro kontrolované vlastnosti

Referenční korektní hodnoty pro vyhodnocování kontrolovaných vlastností jsou získány vyhodnocením referenčního správného displeje a tyto hodnoty naměřených parametrů budou uloženy a označeny za referenční. Každý kontrolovaný displej tedy bude srovnáván s referenčním a na základě odchylky od těchto hodnot rozhodnuto, zda parametry displeje vyhovují nebo ne.

Srovnání s naučeným referenčním displejem umožní snadné přenastavení nástroje pro kontrolu různých typů displejů s různými rozměry mřížky pixelů, velikostí pixelů, počtem řádků či počtem pixelů na řádek.

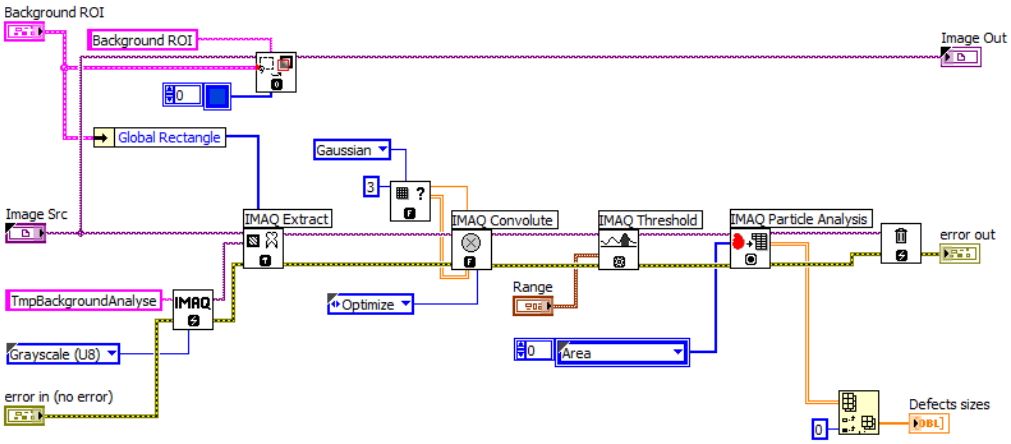
6.4. Analýza pozadí LCD

Kontrola pozadí probíhá na displeji, který nezobrazuje žádné znaky při zapnutém podsvícení. Úkolem kontroly je detekovat jakékoliv mechanické poškození LCD či nečistoty nacházející se na ochranném skle či na podsvíceném pozadí. Algoritmus této operace je navržen tak, aby byla doba kontroly co nejmenší, to umožní při sériové kontrole rychle vyhodnotit nevyhovující displeje.

Algoritmus analýzy pozadí tvoří funkce prahování, ta v obraze hledá objekty, jejichž hodnoty jasu spadají do definovaného rozsahu. Tímto způsobem jsou nalezeny všechny vady, které mají na displeji typu 1602A charakter tmavých oblastí. Na základě počtu a velikosti detekovaných vad je vyhodnoceno, zda vyhovují uživatelsky definovaným požadavkům.

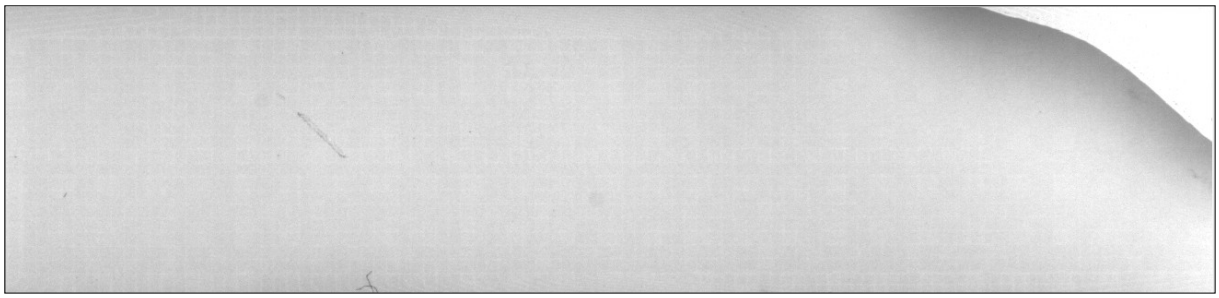
Algoritmus detekce vad na pozadí tvoří tyto dále uvedené operace s obrazem. Nejprve je proveden výřez zájmové oblasti funkcí „IMAQ Extract“, která zkopíruje oblast zájmu. Poté je aplikován Gaussův filtr pro odstranění šumu z obrazu konvoluční funkcí „IMAQ Covolute“ a „IMAQ Get Kernel“, kde je získána příslušná konvoluční maska. Následuje vyprahování vad v obraze funkcí „IMAQ Threshold“, s uživatelky definovaným prahem. Nakonec je provedeno změření jejich velikosti funkcí „IMAQ Particle Analyse“. Časová náročnost algoritmu analýzy pozadí LCD v LabVIEW je přibližně 10ms, na HW popsaném v kapitole 2.4.

Obrázek 43: Vývojový diagram analýzy pozadí LCD



Obrázek 44: Blokový diagram VI pro analýzu pozadí LCD v LabVIEW

Ukázka z detekce vad na pozadí je znázorněna na Obrázek 45 a Obrázek 46. Kontrolovaný displej je v pravém horním rohu poškozen a na pozadí se nachází nečistoty.



Obrázek 45: Analyzovaná oblast pozadí poškozeného LCD

Obrázek 46: Výsledek analýza pozadí poškozeného LCD

[17], [25], [27]

6.5. Analýza pixelů LCD

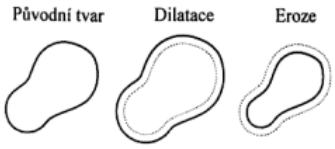
Analýza pixelů probíhá na displeji se zapnutým podsvícením, kdy na všech pozicích je zobrazen plný znak viz Obrázek 28. Cílem kontroly je ověřit funkčnost všech pixelů displeje, jejich velikost a množství světla, které přes ně prochází.

Při implementaci algoritmu pro analýzu funkčnosti pixelů, jsem narazil na problém nerovnoměrnosti podsvícení displeje. Podsvícení LCD displeje tvoří jedna LED dioda, která je umístěna ve výstupku na pravé straně displeje viz Obrázek 27. Toto uspořádání způsobuje jasnější podsvícení u zdroje než na vzdáleném konci displeje. Tato skutečnost mne donutila navrhnout algoritmus tak, aby byl displej kontrolován po částech, to znamená, že před analýzou je potřeba nadefinovat menší oblasti, aby byly porovnávány data vždy ze stejné části displeje, kde hodnota podsvícení je přibližně stejná. Před samotnou detekcí pixelů je na obraz aplikován Gaussův filtr pro odstranění šumu. V následujících podkapitolách jsou popsány vyzkoušené metody pro detekci a analýzu pixelů LCD displeje.

6.5.1. Detekce pixelů hranovým detektorem

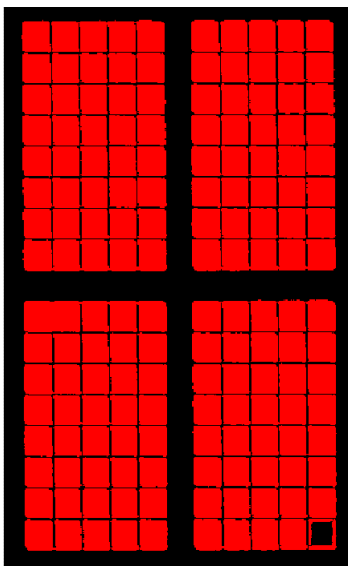
Vzhledem k nerovnoměrnosti podsvícení jsem se zprvu rozhodl pro detekci jednotlivých pixelů použít metodu detekce hran. Hranové detektory, pracují na základě detekce změny jasu mezi jednotlivými body obrazu. Myšlenkou bylo získat obvod jednotlivých pixelů a tento obvod následně vyplnit pro získání masky pro individuální analýzu každého pixelu. Základní funkce „IMAQ Edge Detection“ pro detekci hran (Diferenční, Prewitt, Sobel, Roberts, Sigma) se ukázaly nevhodné, protože docházelo ke spojování hran sousedních pixelů a detekované oblasti tedy neodpovídaly skutečnému rozložení pixelů. Zlepšení výsledku nebylo možné docílit ani prahováním hran či za použití morfologických operací „IMAQ Morphology“.

Morfologické operace se realizují jako relace vstupního obrazu s další bodovou množinou takzvaným strukturním elementem. Operaci si lze představit jako pohyb strukturního elementu po vstupním obraze a vyhodnocení odezvy podle typu operace. Mezi základní morfologické operace se řadí eroze a dilatace. Eroze se používá ke zjednodušení struktur objektů. Odstraňuje objekty jednotkových velikostí a odstraňuje tenké spojnice mezi objekty. Dilatace naopak objekty zvětšuje a zaplňuje v nich díry a zálivy. Ukázka morfologických operací je na Obrázek 47.



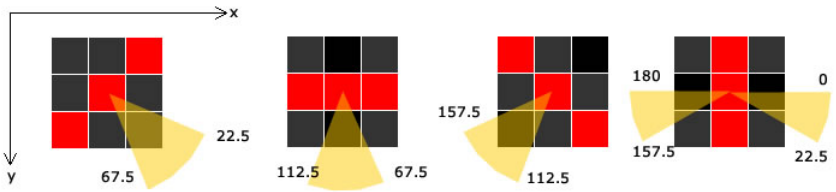
Obrázek 47: Dilatace a eroze

Protože některé sousední nadetekované segmenty se slily v jeden tak, že je nebylo možné erozí oddělit. Ukázka použití diferenčního detektoru hran s vyplnění uzavřených oblastí je na Obrázek 48, často dochází k nežádoucímu spojení detekovaných oblastí nebo neuzavření obvodu.

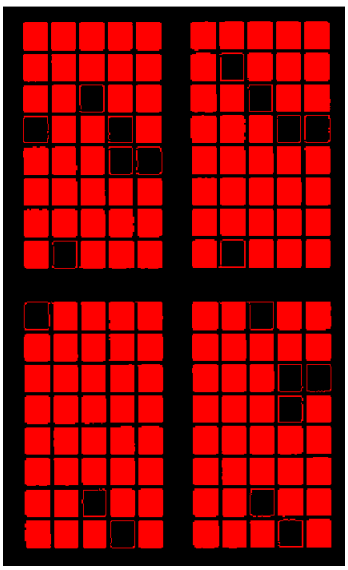


Obrázek 48: Detekce pixelů za použití diferenčního hranového detektoru

Vzhledem k tomu, že základní metody pro detekci hran se ukázaly pro tuto aplikaci nevhodné, rozhodl jsem se vyzkoušet pokročilejší metodu, Cannyho hranový detektor. Tento algoritmus se skládá z několika kroků pro získání co nejlepšího výsledku při detekci hran v dvourozměrném diskrétním obraze. Cannyho hranový detektor se skládá z těchto operací, eliminace šumu gausovým filtrem, detekce hran a jejich směru obvykle Sobelovým operátorem, ztenčení hran na základě jejich směru a nakonec prahování s hysterezí pro odstranění nevýznamných hran. Jeho hlavními vstupními parametry jsou horní a dolní prahová hodnota a velikost masky pro filtrování Gaussovým filtrem. V LabVIEW je tento hranový detektor dostupný ve funkci „IMAQ Canny“.



Obrázek 49: Nalezení směru hrany a směr hledání maxima Cannyho hranovým detektorem



Obrázek 50: Detekce pixelů LCD za použití Cannyho hranového detektoru

Jak je vidět na Obrázek 50, Cannyho hranový detektor lépe nadetekoval tvar pixelů, ale znovu se projevil problém neuzavřených hran. Změnou parametrů nastavení hranového detektoru nebylo možné tuto skutečnost nijak ovlivnit. Pokusil jsem se tento problém vyřešit použitím morfologických operací pro uzavření objektů, ale tato operace nebyla schopna uzavřít všechny hrany a vzhledem k malé vzdálenosti mezi jednotlivými pixely, v některých místech docházelo k nežádoucímu spojování nadetekovaných oblastí. Hranové detektory se tedy pro tento případ aplikace ukázaly jako nevhodné.

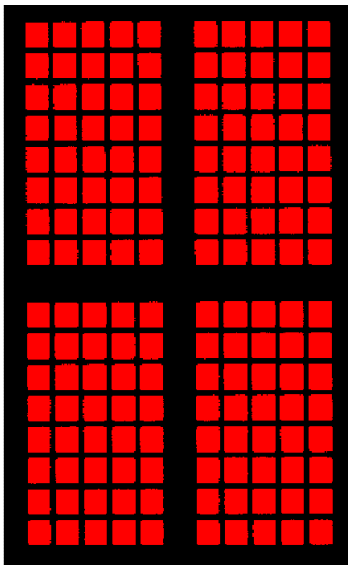
[18], [25]

6.5.2. Detekce pixelů prahováním

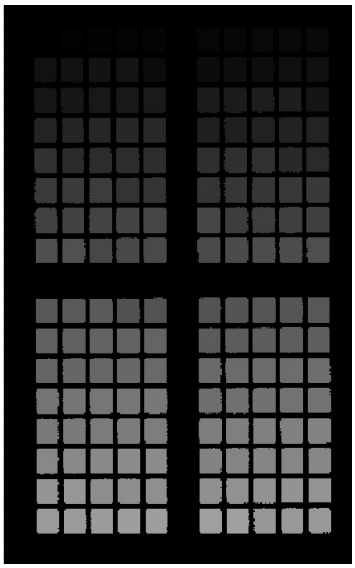
Další metodou, kterou jsem vyzkoušel, pro detekci pixelů bylo prahování. Ruční nastavení prahové hodnoty se ukázalo jako nejlepší řešení, kdy se pro každou část displeje nastaví jiná prahová hodnota. Ukázalo se, že toto nastavování by bylo zbytečně složité při učení referenčních hodnot a při kontrole jiného displeje stejného typu by nemuselo fungovat spolehlivě, proto jsem se rozhodl použít funkce automatického prahování. Automatické prahování vypočítá optimální hodnotu prahování a minimalizuje tím chybu při detekci tmavých oblastí. Automatické prahování v LabVIEW zajišťuje funkce „IMAQ AutoBThreshold 2“. Detekce pixelů automatickým prahováním tmavých objektů je zobrazena na Obrázek 51.

Po vyprahování jednotlivých pixelů jsem tedy získal binární obraz, masku, která udává místa, kde se nacházejí jednotlivé pixely. Aby do analýzy jednotlivých pixelů displeje nebyly zahrnuty i okraje, které by mohly být součástí pozadí a mohly ovlivnit jasovou analýzu, aplikoval jsem na masku morfologickou operaci erozi pomocí funkce „IMAQ Morphology“, čímž se tento obvodový okraj odstraní. Následně je provedena analýza detekovaných objektů funkcí „IMAQ Particle Analyse“ a jejich očíslování hodnotou jasu funkcí „IMAQ Label Particles“.

Vzhledem k tomu, že pracuji s 8bitovým obrázkem, nesmí být v jedné analyzované oblasti více než 255 pixelů displeje, protože by je nebylo možné očíslovat a následně jednotlivě analyzovat. Proto každou část displeje analyzuji zvlášť postupným vyřezávání funkcí „IMAQ Extract“. Řešením by bylo použití 16bitového obrázku, ale při následné jasové analýze by bylo nutné znovu vytvářet 8bitovou masku, protože funkce zpracování obrazu podporují pouze 8bitovou masku. Vzhledem k množství analyzovaných pixelů displeje, kterých je 1280, by zbytečná operace kopírování dat mezi jednotlivými typy obrazu, způsobila nárůst časové náročnosti algoritmu, a to by přesáhlo 1 sekundu jak jsem ověřil testem.



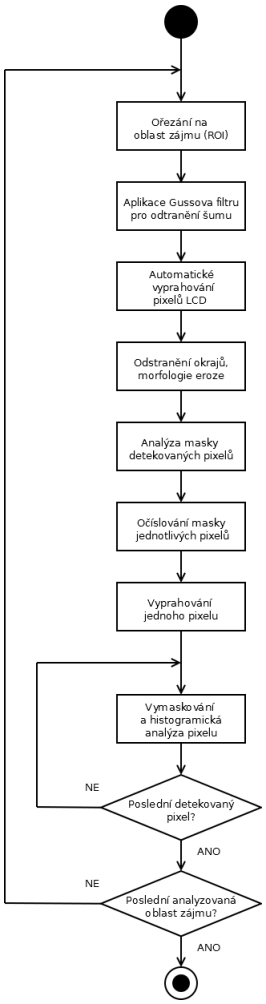
Obrázek 51: Detekce pixelů LCD automatickým prahováním tmavých objektů



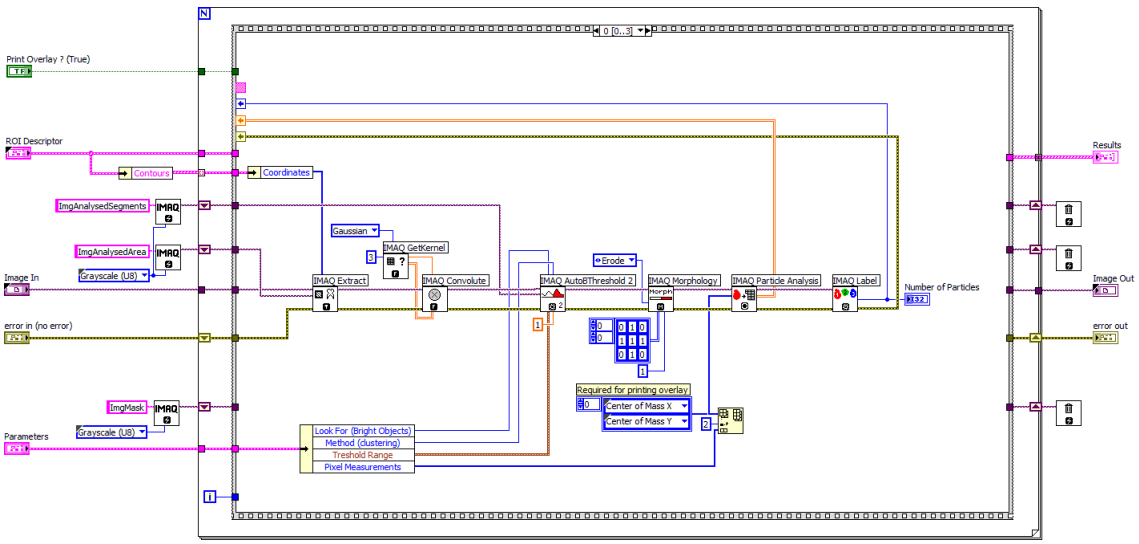
Obrázek 52: Očíslované nadetekované pixely LCD hodnotou jasu

Posledním krokem v analýze je analýza jasu jednotlivých pixelů. K tomu ve zpracování obrazu slouží histogram analýza funkce „IMAQ Histogram“, která provádí statistické výpočty nad oblastí určenou maskou. Z obrazu masky s očíslovanými nadetekovanými pixely LCD jsou postupně vyprahovány jednotlivé pixely „IMAQ Threshold“ a je provedena jejich histogramická analýza. Vývojový diagram analýzy pixelů LCD je na Obrázek 53.

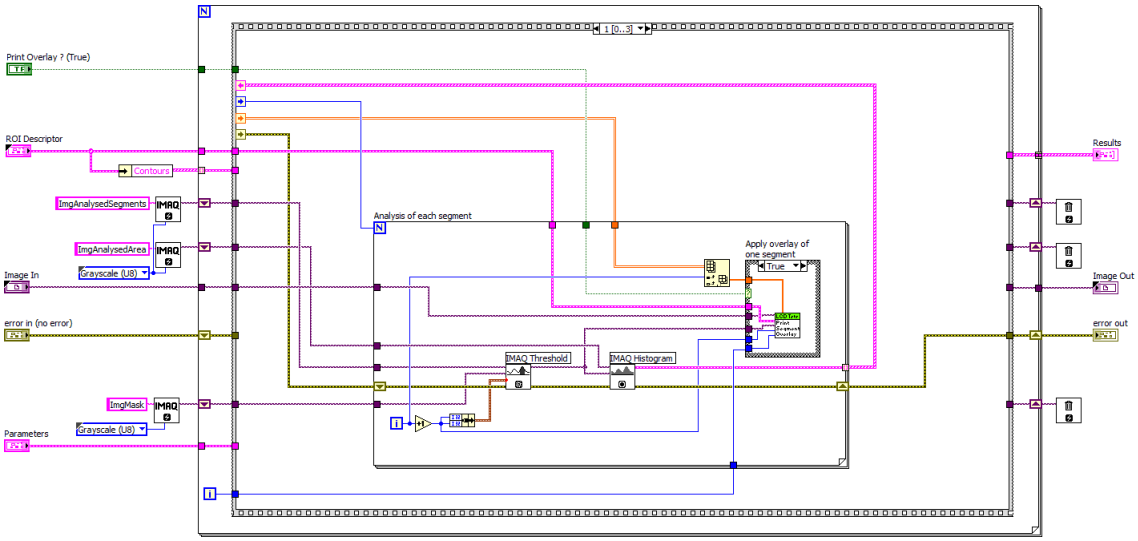
Na Obrázek 54 a Obrázek 54 je zobrazen blokový diagram algoritmu pro analýzu pixelů v LabVIEW. Vstupem algoritmu je analyzovaný snímek LCD displeje spolu s parametry testu a ROI (Oblasti zájmu), ve kterých se nachází analyzované pixely LCD. Výstupem je pak pole obsahující data histogramické analýzy jednotlivých pixelů LCD. Časová náročnost algoritmu analýzy pixelů v LabVIEW se pohybuje okolo 270ms na HW popsaném v kapitole 2.4.



Obrázek 53: Vývojový diagram analýzy pixelů LCD



Obrázek 54: Blokový diagram algoritmu detekce pixelů LCD v LabVIEW



Obrázek 55: Blokový diagram algoritmu jasové analýzy pixelů LCD v LabVIEW

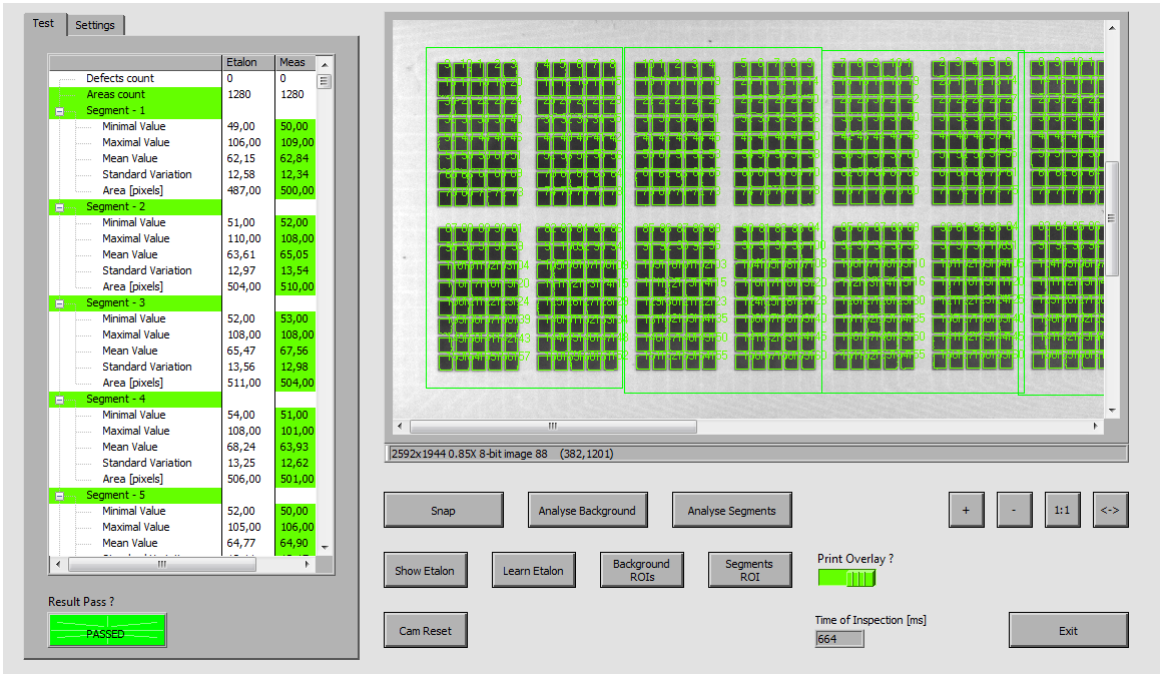
[17], [25], [27]

6.5.3. Nástroj pro kontrolu LCD v LabVIEW

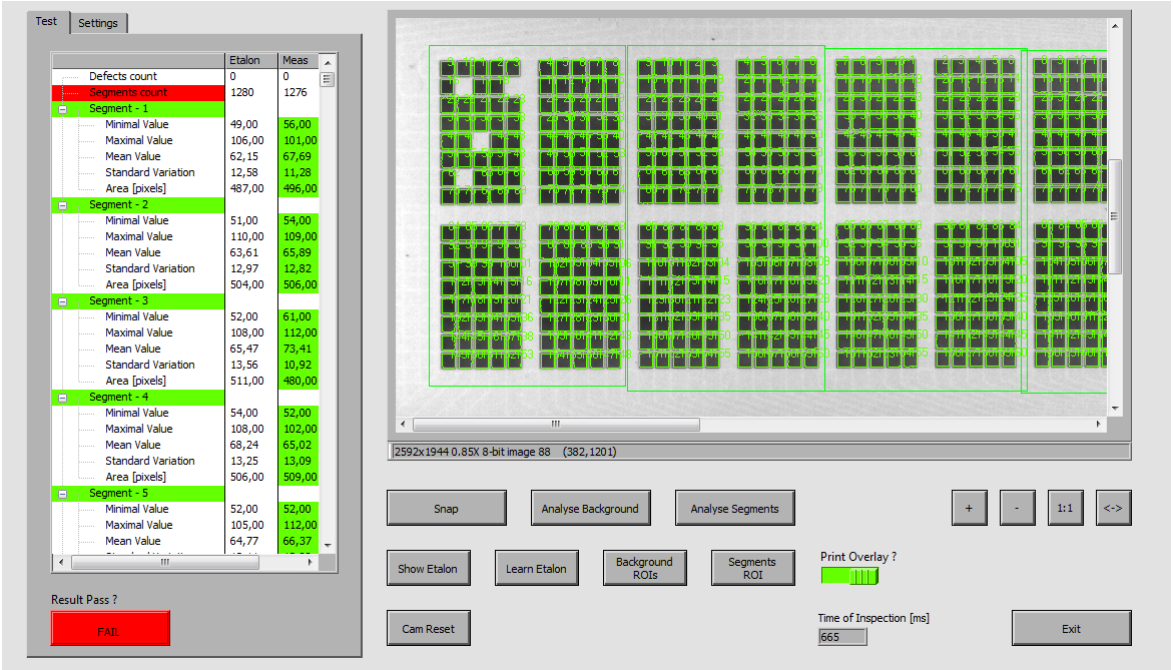
Výše navržené algoritmy pro analýzu LCD jsou použity v hlavní aplikaci, která tvoří výsledný nástroj pro kontrolu monochromatických LCD displejů. Nástroj zastřešuje všechny operace potřebné pro sejmutí obrazu z kamery, komunikaci s LCD displejem, analýzu snímku z kamery, vyhodnocení a vizualizaci výsledků a interakci s uživatelem při nastavování parametrů testu.

Uživatelské rozhraní jsem navrhnul tak, aby bylo co nejjednodušší na ovládání. Vzhled uživatelského rozhraní je na

Obrázek 56. Vlevo se nachází výsledkový strom, ve kterém jsou zobrazeny výsledky celkové analýzy ve srovnání s naučeným vzorem LCD displeje. Vyhovující hodnoty jsou podbarveny zeleně a nevyhovující červeně pro lepší vizuální přehlednost. Vpravo nahoře je okno zobrazující sejmutý snímek, ve kterém lze na žádost vykreslit oblasti nadetekovaných pixelů LCD displeje. Vykreslování lze přepínačem vypnout, protože má pouze vizualizační charakter a zdvojnásobuje čas potřebný pro analyzování LCD. Ve zbylé části aplikace jsou tlačítka pro ovládání celého nástroje.

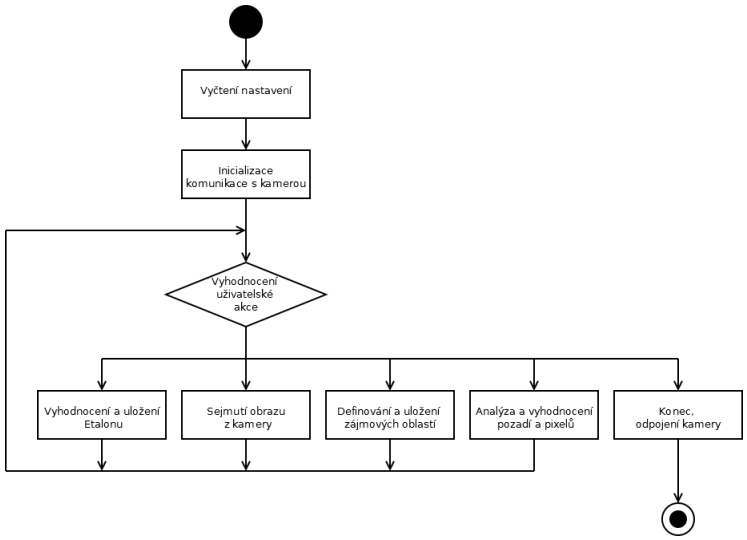


Obrázek 56: Čelní panel nástroje pro kontrolu LCD s vyhodnocením LCD OK

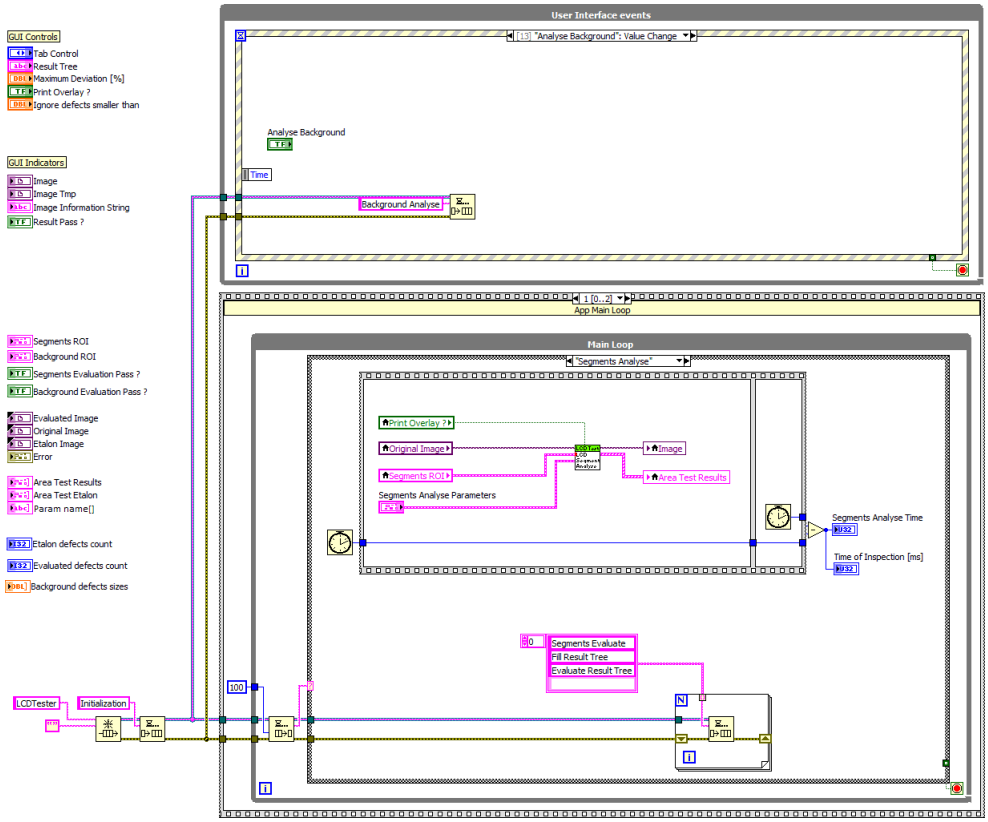


Obrázek 57: Čelní panel nástroje s vyhodnocením LCD NOK

Na Obrázek 58 je vývojový diagram běhu a na Obrázek 59 je blokové schéma nástroje pro testování LCD. Samotnou aplikaci tvoří, stejně jako aplikaci pro komunikaci s LCD, dvě smyčky, horní smyčka slouží pro zachycení událostí na GUI, jako je stisknutí tlačítka na čelním panelu a dolní, která má v sobě výkonný kód. Obě smyčky jsou synchronizovány frontou, do které horní smyčka vloží požadavek a spodní na základě druhu požadavku jej vykoná. Fronta událostí umožňuje naplnění požadavky a realizovat tak funkci stavového automatu.



Obrázek 58: Vývojový diagram běhu nástroje pro kontrolu LCD



Obrázek 59: Blokový diagram nástroje pro testování LCD

Zdrojové kódy algoritmů a nástroje pro testování jsou uloženy na přiloženém CD ve složce LCDTester\LabVIEW. Jedná se o soubory s příponou *.vi.

[1], [17], [16], [18], [25], [27], [30]

6.6. Algoritmy pro kontrolu LCD v C++

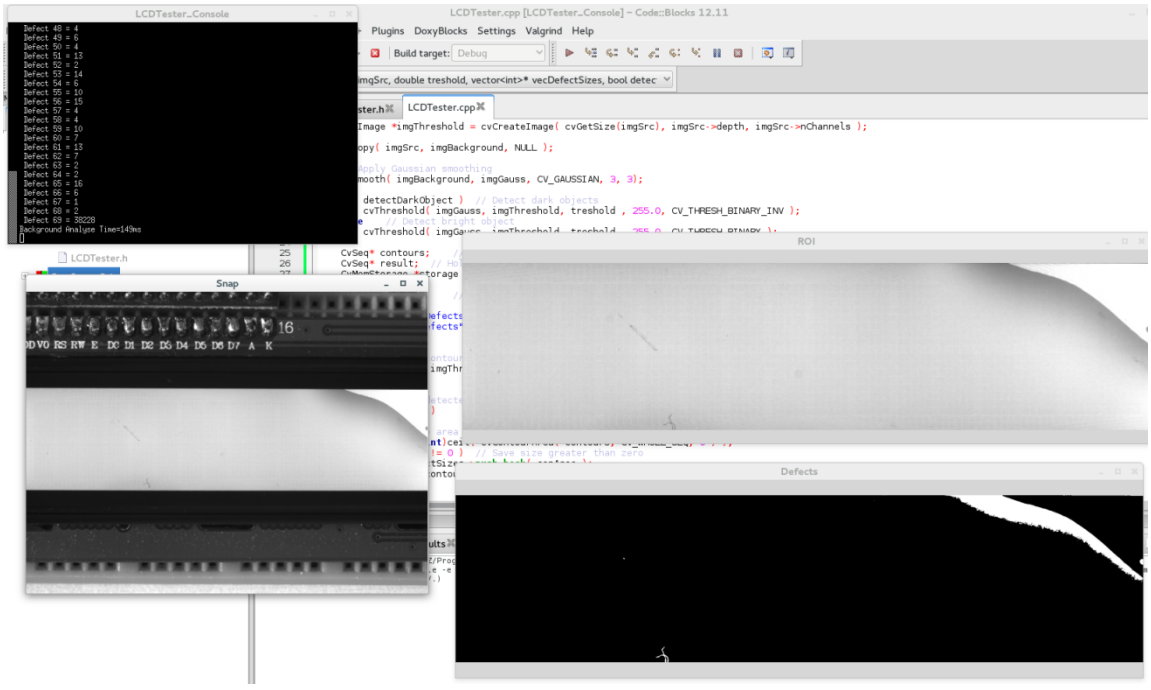
Pro srovnání časové náročnosti algoritmů navržených v LabVIEW, uvedených v předchozích kapitolách, jsem naprogramoval algoritmy v C++ na operačním systému Fedora. Funkce pro zpracování obrazu jsou použity vhodné funkce dostupné z knihovny OpenCV. Pro vyzkoušení a změření rychlosti těchto algoritmů jsem naprogramoval konzolovou aplikaci, ve které jsem využil vizualizačních GUI pro zobrazení výsledků zpracování obrazu. Zdrojové kódy algoritmů analýzy LCD displeje jsou uloženy na přiloženém CD v adresáři LCDTester/C++. Soubory mají příponu *.cpp.

OpenCV nabízí API (Aplication Programing Interface) pro programování v jazyce C i C++, já jsem se rozhodl využít rozhraní C, protože je z mého pohledu přehlednější.

6.6.1. Analýza pozadí

Postup analýzy pozadí je zobrazen vývojovým diagramem v kapitole 6.4 viz Obrázek 43. Nejprve je funkcí „cvCopy“ vykopírována zájmová oblast. Pro odstranění šumu je aplikován gaussův filtr funkcí „cvSmooth“ s parametrem „CV_GAUSSIAN“. Poté jsou funkcí „cvThreshold“ vyprahovány vady na pozadí. Následně se provede nalezení všech objektů funkcí „cvFindCountous“ a postupně je analyzována jejich velikost funkcí „cvContourArea“. Časová náročnost tohoto algoritmu v C++ se pohybuje okolo 7ms na HW popsaném v kapitole 2.4. Analýza pozadí v C++ je uvedena na Obrázek 60.

Ukázka zdrojového kódu funkce pro analýzu pozadí LCD je v Příloha II.



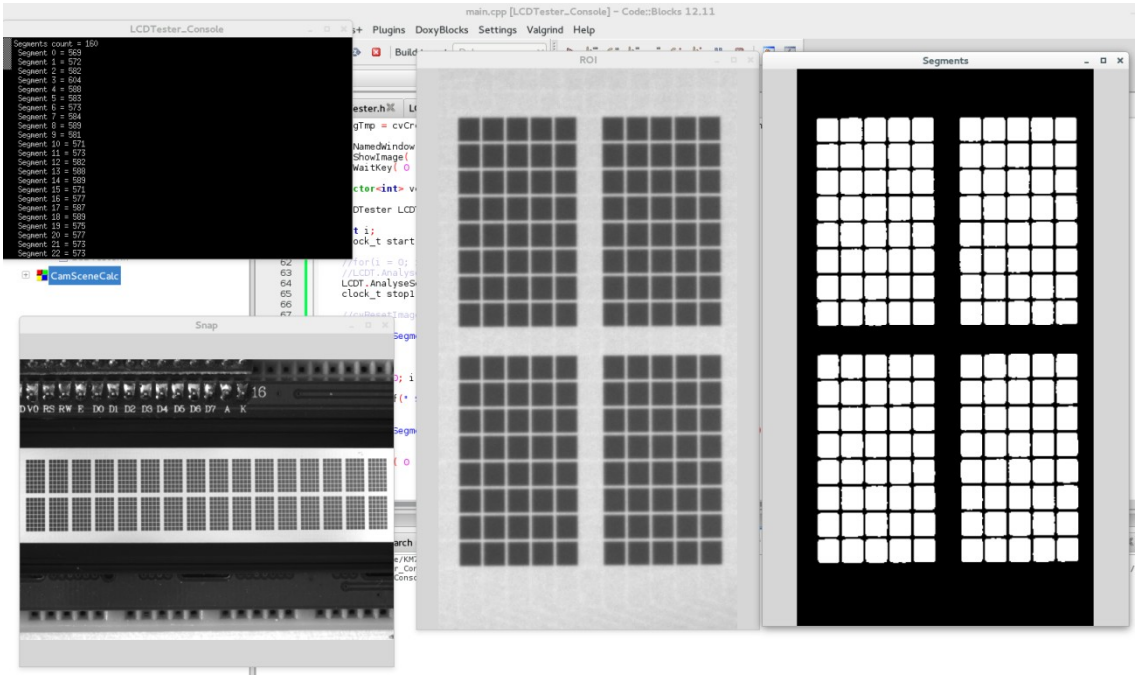
Obrázek 60: Analýza pozadí LCD v C++

[3], [20], [21], [25], [27]

6.6.2. Analýza pixelů

Analýza pixelů displeje v C++ se řídí vývojovým diagramem uvedeným v kapitole 6.5 viz Obrázek 53. Jako první je proveden ořez na zájmovou oblast funkcí „cvCopy“ a aplikován Gaussův filtr funkcí „cvSmooth“ s parametrem „CV_GAUSSIAN“. Poté je aplikováno adaptivním prahování funkcí „cvThreshold“ s parametrem „CV_THRESHOLD_OTSU“ pro získání všech oblastí kde se nacházejí pixely. Funkcí „cvFindContours“ jsou detekovány všechny objekty pro jejich následující postupnou analýzu. Pro každý detekovaný objekt je vytvořena maska za pomocí funkce „cvDrawContours“, která nakreslí analyzovaný objekt do prázdné masky. Nakonec je postupně za pomocí masky provedena histogramická analýza funkcí „cvCalcHist“ a výsledky jsou uloženy pro pozdější vyhodnocení. Algoritmus pro analýzu všech pixelů v C++ dosahuje časové náročnosti 257ms. 7ms na HW popsaném v kapitole 2.4. Analýza jedné z části displeje je uvedena na Obrázek 61.

Ukázka zdrojového kódu funkce pro analýzu pozadí LCD je v Příloha III.



Obrázek 61: Analýza pixelů LCD v C++

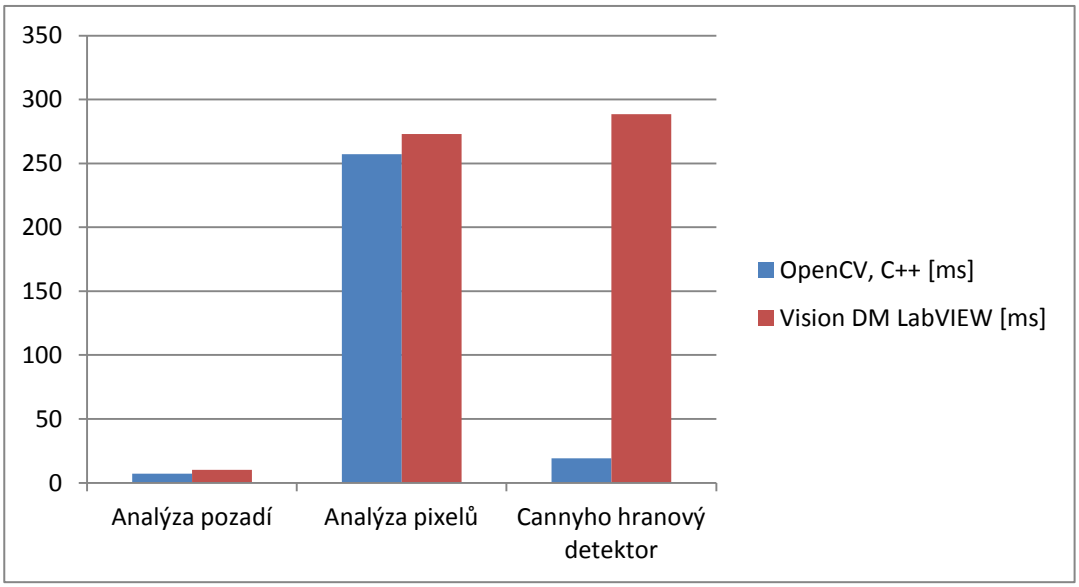
[3], [20], [21], [25], [27]

7. Porovnání rychlosti algoritmů z jednotlivých vývojových prostředí

Po návrhu algoritmů pro kontrolu LCD v jednotlivých vývojových prostředích, jsem vždy provedl změření jejich časové náročnosti pro jejich srovnání. Aby byl výsledek co nejpřesnější, prováděl jsem měření vždy na stejných obrazových datech se stejnými vstupními parametry jednotlivých použitých funkcí. Měření rychlosti jsem proved vždy 10x, vyloučil nejvyšší a nejnižší výsledek a zbylé naměřené výsledky zprůměroval pro získání co nejpřesnějšího výsledku. Naměřené hodnoty časové náročnosti algoritmů z jednotlivých vývojových prostředí viz Tabulka 1. Všechna měření byla provedena na HW popsaném v kapitole 2.4.

Tabulka 1: Srovnání časové náročnosti algoritmů

	Analýza pozadí	Analýza pixelů	Cannyho hranový detektor
OpenCV, C++ [ms]	7,25	257,2	19,25
Vision DM LabVIEW [ms]	10,25	273	288,5



Obrázek 62: Graf srovnání algoritmů

Pro porovnání jsem také přidal do srovnání jednotlivých algoritmů operaci Cannyho hranového detektoru. Na grafu viz Obrázek 62 je vidět vizuální porovnání jednotlivých rychlostí. Algoritmy pro analýzu LCD displeje jsou rychlejší v OpenCV (C++), ale rozdíl od Vision Development Module (LabVIEW) není příliš velký. Dá se tedy konstatovat, že rychlost použitých funkcí pro analýzu LCD je srovnatelná.

Při srovnání algoritmu Cannyho hranového detektoru se ukázal markantní rozdíl v rychlosti 19,25ms v OpenCV oproti 288,5ms ve Vision Development Module. Tento rozdíl je pravděpodobně způsoben rozdílnou implementací a optimalizací toho algoritmu.

8. Závěr

Při návrhu nástroje pro kontrolu monochromatických LCD displejů jsem se seznámil s moderními metodami návrhu aplikací pro strojové vidění, od sestavení snímací scény, návrhu komunikace mezi kontrolovaným displejem a PC, snímání obrazu z kamery, až po návrh samotných vyhodnocovacích algoritmů. Naučil jsem se uplatnit metody analýzy obrazu za použití nástrojů pro strojové vidění ve vývojových prostředích LabVIEW Vision Development module a OpenCV C++.

Navrhnutý nástroj pro kontrolu znakových monochromatických LCD displejů je použitelný pro kontrolu širokého spektra displejů, které se liší počtem zobrazovaných znaků, řádků, různou velikostí matice zobrazující znak či invertovaným zobrazováním (pozadí je černé a pixely propouštějí světlo). Teoreticky by bylo možné tímto nástrojem kontrolovat plně matickové monochromatické LCD displeje nebo segmentové displeje (např. sedmisegmentový displej). Nástroj je snadno rozšířitelný o další analyzační testy.

Při implementaci algoritmů analýzy LCD displeje v jednotlivých programovacích jazycích se ukázalo, že návrh v programovacím jazyce LabVIEW je mnohem rychlejší a intuitivnější než v jazyce C++. LabVIEW poskytuje nástroj Vision Development asistant, ve kterém lze velice rychle navrhnout, otestovat a vygenerovat navržený algoritmus. Implementace algoritmů v C++ s využitím funkcí z OpenCV knihovny není tak jednoduché, ale jak ukázaly testy navržených algoritmů, lepší optimalizace knihovny poskytuje vyšší rychlost při zpracování obrazových dat. V obou programovacích jazycích je možná vlastní implementace funkcí pro zpracování obrazu, ale jazyk C++ je v tomto ohledu flexibilnější, protože umožňuje lépe optimalizovat práci s obrazovými daty. Vzhledem k tomu, že OpenCV je knihovna s otevřeným zdrojovým kódem lze nahlédnout a inspirovat se implementacemi jednotlivých funkcí. V případě implementace v LabVIEW Vision development module je nutné se spolehnout na v něm naimplementované funkce, nebo se spokojit s vlastními funkcemi, které budou vždy pomalejšími. Protože LabVIEW obsahuje prostředky pro volání externích dll knihoven je možné namísto funkcí z Vision Development Module využít rychlejší funkce z OpenCV a spojit tak výhody rychlého vývoje s použitím optimalizovanějších funkcí pro zpracování obrazu.

Součástí práce je také nástroj pro usnadnění návrhu snímací scény, kterým lze snadno vypočítat poslední parametr z rovnice popisující zobrazení objektivem.

9. Literatura

[1] BITTER, Rick, Taqi MOHIUDDIN a Matt NAWROCKI. LabView advanced programming techniques. 2nd ed. Boca Raton: CRC Press, c2007, 499 s. ISBN 08-493-3325-3.

[2] Code::Blocks. [online]. [cit. 2014-05-04]. Dostupné z: http://www.codeblocks.org/

[3] DAVIS, Stephen Randy. *C for dummies*. 5th ed. Indianapolis, IN: Wiley, 2004, p. ISBN 07-645-6852-3.

[4] DOC. ING. JAN ŽÍDEK, CSc. MĚŘENÍ, MĚŘICÍ ŘETĚZEC, KONCEPCE MĚŘICÍCH SYSTÉMŮ [online]. 14.5.2008 [cit. 2014-04-30]. Dostupné z: http://fei1.vsb.cz/stud_mat/K450/Volitelne_Predmety/Mereni_V_Informacnich_A_Komunikacnich_Technologiich/01_tyden/prednaska/P01_Mereni_Merici_retezec_Merici_pristroje.pdf

[5] Fedora 20. [online]. [cit. 2014-05-04]. Dostupné z: http://fedoraproject.org/cs/

[6] GCC, the GNU Compiler Collection: http://gcc.gnu.org/. [online]. [cit. 2014-05-04].

[7] GDM1602A SPECIFICATIONS OF LCD MODULE. [CD]. [cit. 2014-05-04]. gdm1602a-rn-gbs-datasheet-1.pdf

[8] IICLCD. [online]. [cit. 2014-05-04]. Dostupné z: http://arduino-info.wikispaces.com/LCD-Blue-I2C

[9] Kamera Basler acA2500-14gm. [online]. [cit. 2014-05-04]. Dostupné z: http://www.baslerweb.com/products/ace.html?model=170

[10] Komunikace po sériové sběrnici I2C. [online]. [cit. 2014-05-04]. Dostupné z: http://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc/

[11] KOMUNIKAČNÍ ROZHRANÍ VE STROJOVÉM VIDĚNÍ. [online]. [cit. 2014-05-04]. Dostupné z: http://www.visionx.cz/know-how/srovnani-nejpouzivanejsich-komunikacnich-rozhrani-ve-strojovem-videni/

[12] LCD a jejich technologie. [online]. [cit. 2014-05-04]. Dostupné z: http://notebook.cz/clanky/technologie/2013/LCD-1

[13] LCD Interfacing Tutorial: LCD 4-bit Mode Introduction. [online]. [cit. 2014-05-04]. Dostupné z: http://www.8051projects.net/lcd-interfacing/lcd-4-bit.php

[14] NI LabVIEW. [online]. [cit. 2014-05-04]. Dostupné z: http://czech.ni.com/labview

[15] NI USB-8451. [online]. [cit. 2014-05-04]. Dostupné z: http://sine.ni.com/nips/cds/view/p/lang/cs/nid/202368

[16] NI Vision Acquisition Software. [online]. [cit. 2014-05-04]. Dostupné z: http://sine.ni.com/nips/cds/view/p/lang/cs/nid/12892

[17] NI Vision Assistant Tutorial, VA_Tutorial.PDF, součást instalace LabVIEW 2013

[18] NI Vision Development Module. [online]. [cit. 2014-05-04]. Dostupné z: http://www.ni.com/labview/vision/

[19] Objektivy pro strojové vidění. [online]. [cit. 2014-05-04]. Dostupné z: http://www.analyza-obrazu.cz/objektivy/

[20] OpenCV Documentation. [online]. [cit. 2014-05-04]. Dostupné z: http://docs.opencv.org/

[21] OpenCV Tutorial C++. [online]. [cit. 2014-05-04]. Dostupné z: http://opencv-srf.blogspot.cz/

[22] OpenCV. [online]. [cit. 2014-05-04]. Dostupné z: http://opencv.org/

[23] PCF8574 Remote 8-bit I/O expander for I2C-bus. [CD]. [cit. 2014-05-04]. PCF8574.pdf

[24] Podrobný průvodce objektivy. [online]. [cit. 2014-05-04]. Dostupné z: http://technet.idnes.cz/co-musi-umet-kazdy-objektiv-aby-fotky-staly-za-to-podrobny-pruvodce-11m-/tec_foto.aspx?c=A071108_120848_tec_foto_jlb

[25] SOJKA, Eduard. *Digitální zpracování a analýza obrazů* [online]. 1. vyd. Ostrava: VŠB - Technická univerzita, 2000, 133 s. [cit. 2014-05-04]. ISBN 80-707-8746-5.

[26] Strojové vidění I: Principy a charakteristiky. [online]. [cit. 2014-05-04]. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=36550

[27] Strojové vidění II: Úlohy, nástroje a algoritmy. [online]. [cit. 2014-05-04]. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=36676

[28] Strojové vidění III: Kamery a jejich části. [online]. [cit. 2014-05-04]. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=36925

[29] Telecentric lenses tutorial. [online]. [cit. 2014-05-04]. Dostupné z: http://www.opto-engineering.com/resources/telecentric-lenses-tutorial

[30] VLACH, Jaroslav, Josef HAVLÍČEK a Martin VLACH. *Začínáme s LabVIEW*. 1. vyd. Ilustrace Viktorie Vlachová. Praha: BEN - technická literatura, 2008, 247 s. ISBN 978-80-7300-245-9.

[31] WxSmith. [online]. [cit. 2014-05-04]. Dostupné z: http://wiki.codeblocks.org/index.php?title=WxSmith_tutorials

[32] WxWidgets Documentation. [online]. [cit. 2014-05-04]. Dostupné z: http://www.wxwidgets.org/docs/

[33] WxWidgets. [online]. [cit. 2014-05-04]. Dostupné z: https://www.wxwidgets.org/

10. Seznam obrázků

Obrázek 1: LCD Displej 1602A 2
Obrázek 2: Řadič LCD displeje 2
Obrázek 3: Princip průchodu světla pixelem LCD 3
Obrázek 4: Převodník IICLCD 4
Obrázek 5: Převodník NI USB-8451 4
Obrázek 6: Blokové schéma připojení LCD k PC 4
Obrázek 7: Kamera s připevněným objektivem 7
Obrázek 8: Zobrazení předmětu v kameře objektivem 7
Obrázek 9: Lamelová clona objektivu 8
Obrázek 10: Hloubka ostrosti 9
Obrázek 11: Srovnání zobrazení (nalevo) klasickým a (napravo) telecentrickým objektivem 9
Obrázek 12: Zobrazení objektu (nahoře) klasickým a (dole) telecentrickým objektivem 10
Obrázek 13: Ukázka blokového diagramu VI 11
Obrázek 14: Ukázka čelního panelu VI 12
Obrázek 15: Logo LabVIEW 12
Obrázek 16: Ukázka funkcí pro strojové vidění v LabVIEW 13
Obrázek 17: Ukázka návrhu algoritmu strojového vidění v NI Vision Assistant 14
Obrázek 18: Vývojové prostředí Code::Blocks 15
Obrázek 19: Logo Code::Blocks 15
Obrázek 20: Logo CNU GCC 15
Obrázek 21: Vývojové prostředí Code::Blocks s grafickým modulem wxSmith 16
Obrázek 22: Logo wxWidgets 16
Obrázek 23: Logo OpenCv 17
Obrázek 24: Nástroj pro výpočet pracovní vzdálenosti objektivu 18
Obrázek 25: Snímací scéna 19
Obrázek 26: Snímek z kamery pro displej bez zobrazených znaků 20
Obrázek 27: LCD displej se zapnutým podsvícením 20
Obrázek 28: LCD displej zobrazující plné znaky na všech pozicích 21
Obrázek 29: Schéma propojení displeje s převodníky 21
Obrázek 30: Základní znaková sada LCD displeje 22
Obrázek 31: Vývojový diagram aplikace pro komunikaci s LCD displejem 23
Obrázek 32: VI I2CLCD Server 23

Obrázek 33: Čelní panel Aplikace pro komunikaci s LCD displejem 23
Obrázek 34: Blokový diagram aplikace I2CLCD 24
Obrázek 35: Podsvícený LCD pro analýzu pozadí 24
Obrázek 36: LCD bez znaků s vadnými pixely 25
Obrázek 37: LCD bez vad zobrazení 25
Obrázek 38: LCD s vadnými pixely 25
Obrázek 39: LCD s vadným celým řádkem pixelů 25
Obrázek 40: Mechanicky poškozený LCD 26
Obrázek 41: LCD s hrubými nečistotami 26
Obrázek 42: Vývojový diagram kontroly LCD 28
Obrázek 43: Vývojový diagram analýzy pozadí LCD 29
Obrázek 44: Blokový diagram VI pro analýzu pozadí LCD v LabVIEW 29
Obrázek 45: Analyzovaná oblast pozadí poškozeného LCD 30
Obrázek 46: Výsledek analýza pozadí poškozeného LCD 30
Obrázek 47: Dilatace a eroze 31
Obrázek 48: Detekce pixelů za použití diferenčního hranového detektoru 31
Obrázek 49: Nalezení směru hrany a směr hledání maxima Cannyho hranovým detektorem 32
Obrázek 50: Detekce pixelů LCD za použití Cannyho hranového detektoru 32
Obrázek 51: Detekce pixelů LCD automatickým prahováním tmavých objektů 33
Obrázek 52: Očíslované nadetekované pixely LCD hodnotou jasu 34
Obrázek 53: Vývojový diagram analýzy pixelů LCD 35
Obrázek 54: Blokový diagram algoritmu detekce pixelů LCD v LabVIEW 35
Obrázek 55: Blokový diagram algoritmu jasové analýzy pixelů LCD v LabVIEW 36
Obrázek 56: Čelní panel nástroje pro kontrolu LCD s vyhodnocením LCD OK 37
Obrázek 57: Čelní panel nástroje s vyhodnocením LCD NOK 37
Obrázek 58: Vývojový diagram běhu nástroje pro kontrolu LCD 38
Obrázek 59: Blokový diagram nástroje pro testování LCD 38
Obrázek 60: Analýza pozadí LCD v C++ 39
Obrázek 61: Analýza pixelů LCD v C++ 40
Obrázek 62: Graf srovnání algoritmů 41

11. Seznam tabulek

Tabulka 1: Srovnání časové náročnosti algoritmů 41

12. Seznam rovnic

Rovnice 1: Zobrazení objektivem 7

Rovnice 2: Výpočet ohniskové vzdálenosti objektivu 18

13. Seznam příloh

Příloha I: Ukázka textového souboru obsahující typy senzorů a jejich rozměry 48

Příloha II: Zdrojový kód funkce analýzy pozadí LCD v C++..... 49

Příloha III: Zdrojový kód funkce pro analýzu pixelů LCD v C++..... 50

14. Obsah přiloženého CD

\DiplomovaPraceDok	- Diplomová práce v elektronické podobě
\IICLCD	- Nástroj pro komunikaci s LCD
\LCDTester\LabVIEW	- Nástroj pro kontrolu LCD v LabVIEW
\LCDTester\C++	- Algoritmy analýzy LCD v C++
\CamSceneCalc	- Nástroj pro výpočet pracovní vzdálenosti objektivu v C++

Přílohy:

#Type	Width (mm)	Height (mm)
1/4"	3,2	2,4
1/3,6"	4	3
1/3,2"	4,54	3,42
1/3"	4,8	3,6
1/2,7"	5,37	4,04
1/2,5"	5,76	4,29
1/2"	6,4	4,8
1/1,8"	7,18	5,32
1/1,7"	7,6	5,7
1/1,6"	8,08	6,01
2/3"	8,8	6,6
1"	12,8	9,6
4/3"	17,3	13
1,5"	18,7	14
APS-C	23,6	15,6
Full-frame35mm	36	24
UI-1240SE	6,78	5,43

Příloha I: Ukázka textového souboru obsahující typy senzorů a jejich rozměry

```
int LCDTester::AnalyseBackground( IplImage* imgSrc, double treshold, vector<int>* vecDefectSizes, bool detectDarkObject )
{
    IplImage *imgBackground = cvCreateImage( cvGetSize(imgSrc), imgSrc->depth, imgSrc->nChannels );
    IplImage *imgGauss = cvCreateImage( cvGetSize(imgSrc), imgSrc->depth, imgSrc->nChannels );
    IplImage *imgThreshold = cvCreateImage( cvGetSize(imgSrc), imgSrc->depth, imgSrc->nChannels );
    cvCopy( imgSrc, imgBackground, NULL );  /// Copy ROI from image
    cvSmooth( imgBackground, imgGauss, CV_GAUSSIAN, 3, 3);  /// Apply Gaussian smoothing
    if( detectDarkObject )  /// Detect dark objects
        cvThreshold( imgGauss, imgThreshold, treshold , 255.0, CV_THRESH_BINARY_INV );
    else    /// Detect bright object
        cvThreshold( imgGauss, imgThreshold, treshold , 255.0, CV_THRESH_BINARY );
    CvSeq* contours;    /// Hold the pointer to a contour in the memory block
    CvSeq* result;  /// Hold sequence of points of a contour
    CvMemStorage *storage = cvCreateMemStorage(0);  /// Storage area for all contours
    int conArea = 0;    /// Area size of detected contour
    /// Finding all contours in the image
    cvFindContours( imgThreshold, storage, &contours, sizeof(CvContour), CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE, cvPoint( 0,
0 ) );
    /// Analyse all detected contours
    while( contours )
    {
        /// Calculate area size
        conArea = (int)ceil( cvContourArea( contours, CV_WHOLE_SEQ, 0 ) );
        if( conArea != 0 )  /// Save size greater than zero
            vecDefectSizes->push_back( conArea );
        contours = contours->h_next;
    }
    /// Release memory
    cvReleaseImage(&imgBackground);
    cvReleaseImage(&imgGauss);
    cvReleaseImage(&imgThreshold);
    return 1;
}
```

Příloha II: Zdrojový kód funkce analýzy pozadí LCD v C++

```
int LCDTester::AnalyseSegments( IplImage* imgSrc, double treshold, vector<HistogramData>* vecHistData, bool detectDarkObject )
{
    IplImage *imgSegments = cvCreateImage( cvGetSize(imgSrc), imgSrc->depth, imgSrc->nChannels );
    IplImage *imgGauss = cvCreateImage( cvGetSize(imgSrc), imgSrc->depth, imgSrc->nChannels );
    IplImage *imgAutoThreshold = cvCreateImage( cvGetSize(imgSrc), imgSrc->depth, imgSrc->nChannels );
    IplImage *imgSegmentMask = cvCreateImage( cvGetSize(imgSrc), imgSrc->depth, imgSrc->nChannels );
    cvCopy( imgSrc, imgSegments, NULL );    /// Copy ROI from image
    cvSmooth( imgSegments, imgGauss, CV_GAUSSIAN, 3, 3);    /// Apply Gaussian smoothing
    if( detectDarkObject )  /// Detect dark objects
        cvThreshold( imgGauss, imgAutoThreshold, treshold , 255.0, CV_THRESH_BINARY_INV + CV_THRESH_OTSU );
    else    /// Detect bright object
        cvThreshold( imgGauss, imgAutoThreshold, treshold , 255.0, CV_THRESH_BINARY + CV_THRESH_OTSU );
    CvSeq* contours;    /// Hold the pointer to a contour in the memory block
    CvSeq* result;  /// Hold sequence of points of a contour
    CvMemStorage *storage = cvCreateMemStorage(0);  /// Storage area for all contours
    int conArea = 0;    /// Area size of detected contour
    /// Finding all contours in the image
    cvFindContours( imgAutoThreshold, storage, &contours, sizeof(CvContour), CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE, cvPoint( 0, 0 ) );
    float histMin, histMax, histMean, value = 0.0;
    float range_0[]={0,256};
    float *ranges[]={range_0};
    int histSize = 256;
    CvHistogram* hist = cvCreateHist(1, &histSize, CV_HIST_ARRAY, ranges, 1); /// Histogram
    HistogramData histData; /// Histogram Data
    /// Analyse all detected contours
    while( contours )
    {
        /// Calculate area size
        conArea = (int)ceil( cvContourArea( contours, CV_WHOLE_SEQ, 0 ) );
        if( conArea != 0 )  /// Save size greater than zero
        {
            cvZero( imgSegmentMask );   /// Clear Mask
            /// Draw mask of segment
            cvDrawContours( imgSegmentMask, contours, cvScalar(255), cvScalar(255), 0, -1, 8, cvPoint(0,0) );
            /// Calculate histogram of segment
            cvCalcHist( &imgGauss, hist, 0, imgSegmentMask);
            /// Calculate histogram data
            CalculateHistData( hist, &histData);
            histData.area = conArea;    /// Save segment size
            vecHistData->push_back( histData ); /// Save histogram data
        }
        contours = contours->h_next;
    }
    /// Release memory
    cvReleaseImage(&imgSegments);
    cvReleaseImage(&imgGauss);
    cvReleaseImage(&imgAutoThreshold);
    return 1;
}
```

Příloha III: Zdrojový kód funkce pro analýzu pixelů LCD v C++